UNIVERSIDADE FEDERAL DA BAHIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MECATRÔNICA
MESTRADO EM MECATRÔNICA

ANTONIO CLÁUDIO LOPES DE ARAÚJO

SCAE: UMA ABORDAGEM BASEADA EM AGENTES INTELIGENTES PARA GERENCIAMENTO E CONTROLE DE CAMPOS DE PETRÓLEO

Salvador
2011

ANTONIO CLÁUDIO LOPES DE ARAÚJO

SCAE: UMA ABORDAGEM BASEADA EM AGENTES INTELIGENTES PARA GERENCIAMENTO E CONTROLE DE CAMPOS DE PETRÓLEO

Dissertação apresentada ao Programa de Pós-Graduação em Mecatrônica da Universidade Federal da Bahia, como requisito para obtenção do grau de Mestre em Mecatrônica.

Orientador:
Prof. Herman Augusto Lepikson, Dr. Eng.

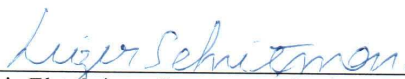
Salvador
2011

A663 Araújo, Antonio Claudio Lopes de
SCAE – uma abordagem baseada em agentes inteligentes para o gerenciamento e controle de campos de petróleo / Antonio Cláudio Lopes de Araújo. – Salvador, 2011.
132 f. : il. color.

Orientador: Prof. Doutor Herman Augusto Lepikson

Dissertação (mestrado) – Universidade Federal da Bahia. Escola Politécnica, 2011.

1. Agentes inteligentes (Software). 2. Indústria petrolífera – controle de produção. 3. Software - Desenvolvimento. I. Lepikson, Herman Augusto. II. Universidade Federal da Bahia. III. Título.

CDD.: 658.5

TERMO DE APROVAÇÃO

ANTONIO CLÁUDIO LOPES DE ARAÚJO

SCAE: UMA ABORDAGEM BASEADA EM AGENTES INTELIGENTES PARA GERENCIAMENTO E CONTROLE DE CAMPOS DE PETRÓLEO

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Mecatrônica, Universidade Federal da Bahia - UFBA, pela seguinte banca examinadora:

Herman Augusto Lepikson – Orientador ______________________
Doutor em Engenharia Mecânica, Universidade Federal de Santa Catarina (UFSC)
Universidade Federal da Bahia

Leizer Schnitman ______________________
Doutor em Engenharia Eletrônica e Computação, Instituto Tecnológico de Aeronáutica (ITA)
Universidade Federal da Bahia

Mário César Mello Massa de Campos______________________
Doutor em Engenharia Química, Ecole Central Paris, França
PETROBRAS - CENPES

José Francisco dos Santos Corrêa______________________
Mestre em Engenharia de Petróleo, Universidade Estadual de Campinas (UNICAMP)
PETROBRAS – UO/BA

Salvador, 25 de março de 2011.

A minha filha Rebeca.

Mesmo ainda sem entender, seu apoio foi fundamental.

AGRADECIMENTOS

Quando resolvemos fazer algo que parece grandioso, ao menos para nós mesmos, nos deparamos com nossas limitações e vemos quanto somos dependentes, não apenas da natureza que nos rodeia, mas principalmente das pessoas. Portanto, essa caminhada com cara e jeito de expedição ao desconhecido não poderia ser realizada sem o apoio, silencioso ou não, dos que me rodeiam. Dessa forma aqui vai uma tentativa de homenagear aqueles que estiveram comigo ou passaram por mim durante a caminhada.

Ao professor Herman Augusto Lepikson, por ter acreditado nesse trabalho (e em mim) quando nem eu acreditava ainda.

Aos membros da banca pela disposição em contribuir para o enriquecimento desse trabalho.

Aos professores do PPGM, pelos ensinamentos.

À Petrobras e à Rockwell Automation pelo apoio técnico e financeiro que possibilitou a realização deste trabalho.

Aos amigos do LEA, pela companhia durante a pesquisa, em especial aos amigos Cícero, Lairton e Tiago fundamentais em diversos momentos.

Ao pessoal do CTAI que tão atenciosamente trabalhou para que tenhamos as condições necessárias à realização da pesquisa.

A Luciana, pela companhia na realização dos experimentos e em alguns momentos difíceis no trabalho escrito.

Aos colegas da Estácio – FIB Helder, Marcos Lapa e Carlos Palma (a lista é grande, resumi), pelo constante incentivo.

Aos amigos com quem compartilhei as idéias aqui expostas, em especial a Marcus Vinícius e Mário Jorge.

A minha amiga Tatiana Dias, por dividir comigo os primeiros desafios no PPGM.

Aos meus pais, responsáveis pelo que sou hoje.

Aos meus irmãos pelo amor, carinho e compreensão.

A minha esposa Débora, guerreira que suportou a pior parte desse trabalho (eu), e a minha filha Rebeca, por ter me deixado usar o computador algumas vezes, mas principalmente por estar me mostrando o que é a vida e como ela deve ser vivida.

A Deus, pois sem Ele "...nada do que foi feito se fez.", portanto toda honra e toda glória sejam dadas a Tí.

Se caminhar sozinho, quem vai te levantar quando cair?

RESUMO

Diversos sistemas existentes, seja na natureza, sejam nas sociedades humanas apresentam a característica peculiar de serem formados por subsistemas ou indivíduos que cooperam entre si para a realização de suas tarefas, e consequentemente alcançarem um objetivo global. Exemplos desses sistemas vão de uma sociedade de abelhas a uma organização financeira, de uma comunidade animal a um sistema de manufatura, entre tantos outros. Ao se observar um campo de produção de petróleo, por exemplo, é possível perceber a natureza distribuída e colaborativa desse tipo de sistema, onde diversos poços (de produção ou injeção), estações de coleta e tratamento de óleo, entre outros, distribuídos geograficamente, têm uma distribuição que naturalmente os induziria a cooperar constantemente para que os objetivos de produção sejam atingidos. Apesar disso, as atuais arquiteturas para operação de um campo de produção utilizam uma estrutura centralizada para o seu gerenciamento. As oportunidades advindas da automação, do aumento da capacidade de processamento dos sistemas computacionais e da diminuição do custo das plataformas de hardware têm, por sua vez, ensejado oportunidades para aperfeiçoar o modo de produção dos campos de petróleo. Os sistemas de manufatura por sua vez, têm sido cada vez mais influenciados pelos sistemas multiagentes, um novo paradigma para o desenvolvimento de software que tem no agente o seu elemento principal. Alguns dos benefícios deste paradigma são: maior poder de atuação, por conta comportamento autônomo e inteligente, maior capacidade de lidar com as falhas nos sistemas de computação, por conta da descentralização do controle, e maior flexibilidade por conta da estrutura dinâmica dos agentes. Este trabalho apresenta um modelo baseado em sistema multiagente para o gerenciamento e controle de um campo de petróleo, cuja proposta visa a dotar cada unidade do sistema produtivo com a capacidade de análise local de suas condições baseada no conhecimento especialista embarcado, de comunicação e de negociação com outras unidades a fim de estabelecer decisões com base em uma visão global do sistema que permita consequentemente, melhorar a sua operação.

Palavras chave: agente; sistemas multiagentes; sistemas industriais; campos de produção de petróleo.

ABSTRACT

Several existing systems, whether in nature, whether in companies human exhibit the characteristic of being formed by individuals or subsystems that cooperate to carry out their tasks, and thus achieve a global goal. Examples these systems range from a society of bees to an organization financial, community animal to a manufacturing system, among many others. By observing a field of oil production, by example, it is possible to realize the distributed and collaborative nature of this type of system, where several wells (production or injection), stations collection and processing of oil, among others, distributed geographically, have a distribution that naturally induce cooperate constantly so that production targets are achieved. Despite that (Nevertheless), current architectures for operation of a field production using a centralized structure for management. The opportunities resulting from automation, increasing the capacity of processing computer systems and reducing the cost of hardware platforms have, in turn, engaged opportunities for improve the mode of production from oil fields. Systems manufacturing in turn, have been increasingly influenced by multiagent systems, a new paradigm for the development of software agent that has at its core element. Some of benefits of this paradigm are: greater power to act on behalf autonomous and intelligent behavior, greater ability to deal with failures in computing systems, due to the decentralization of control, and increased flexibility due to the dynamic structure of agents. This paper presents a model based on multi-agent system for the management and control of an oil field, whose proposal aims to provide each unit of the production system with the ability to analysis of local conditions based on expert knowledge board, communication and negotiation with other units in order to establishing decisions based on an global overview of the system to consequently, improve its operation.

Keywords: agent, multiagent systems, industrial systems, oil production fields.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AAC	Agente de Ajuste do Controlador
AAL	Agente de Análise Local
ACS	*Autonomous Cooperative System*
AIU	Agente de Interface do Usuário
ANE	Agente de Negociação
POA	Programação Orientada a Agentes
ASI	Agente de Sincronização
AVC	Agente de Verificação Continuada
BC	Base de Conhecimento
BM	Bombeio Mecânico
CAD	Desenho Assistido por Computador (*Computer-Aided Design*)
CDF	Cartas Dinamométricas de Fundo
CDS	Cartas Dinamométricas de Superfície
CLP	Controlador Lógico Programável
COP	Central de Operações da Produção
CPM	Ciclos por Minuto
CR	Camada Reativa
ECO	Estação de coleta
EIAs	Estações de Injeção de Água
ETO	Estação de tratamento
FSM	Máquina de Estados Finitos (*Finite State Machine*)
IDL	*Interface Definition Language*
IHC	Interfaces Humano-Computador
J2ME	*Java 2 Micro Edition*
JADE	*Java Agent Development Enviroment*
JICP	*Jade Inter Container Protocol*
LEA	Laboratório de Elevação Artificial
LEAP	*Lightweight Extensible Agent Platform*
MAC	Módulo de Atualização de Controle
MAL	Módulo de Análise Local
MC	Módulo Cognitivo

MGE	Módulo Gerenciador de Eventos
MGR	Módulo de Gerenciamento Remoto
MNE	Módulo de Negociação
MSI	Módulo de Sincronização
MTE	Módulo de Tratamento de Exceção
MVC	Módulo de Verificação Continuada
OPC	*OLE Process for Control*
PC	*Personal Computer*
PDA	*Personal Digital Assistant*
PLA	Plataforma Local de Agentes
POO	Paradigma Orientado a Objetos
SBM	Sistema de Bombeio Mecânico
SD	Sistemas Distribuídos
SIA	Sistemas Industriais Automatizados
GCAD	Controle Autônomo e Distribuído para Sistemas Industriais Automatizados
SMA	Sistemas Multiagentes
UB	Unidade de Bombeio
UFBA	Universidade Federal de Bahia
UML	*Unified Modeling Language*
UP	Unidade da Produção
VSD	Variador de Frequência (*Variable Speed Drive*)

SUMÁRIO

1 INTRODUÇÃO

Diversos sistemas existentes, seja na natureza, seja nas sociedades humanas apresentam a característica peculiar de serem formados por subsistemas ou indivíduos que cooperam entre si para a realização de suas tarefas, e consequentemente alcançarem um objetivo global. Exemplos desses sistemas vão, dentre tantos outros, de uma sociedade de abelhas a uma organização financeira, de uma comunidade animal a um sistema de manufatura.

Os sistemas de manufatura são, em geral, caracterizados por se apresentarem como diversos subsistemas interrelacionados. Ao se observar um campo de produção de petróleo, por exemplo, é possível perceber a natureza distribuída e colaborativa desse tipo de sistema, onde diversos poços (de produção ou injeção), estações de coleta e tratamento de óleo distribuídos geograficamente, cooperam constantemente para que os objetivos de produção sejam atingidos.

Apesar disso, as atuais arquiteturas para operação de um campo de produção utilizam uma estrutura cliente-servidor, na qual o dispositivo que embarca a lógica de controle de uma Unidade da Produção (UP), em geral um Controlador Lógico Programável (CLP), envia periodicamente os valores das variáveis do poço para a Central de Operações da Produção (COP), onde sistemas supervisórios auxiliam os operadores do campo na identificação das condições de funcionamento de cada poço.

Ao perceber alguma anormalidade no funcionamento de uma UP, o operador identifica quais ações podem ser tomadas para ajustar a operação e, caso algum ajuste necessite ser feito, ele altera remotamente a tabela de dados do CLP.

A comunicação entre o controlador e a COP é feita por uma rede de comunicação no padrão mestre-escravo, utilizando, para isso, radio-modens (Figura 1). Nesse modelo, só é permitida a comunicação com a COP de um controlador por vez, pois este utiliza janelas de comunicação baseadas em intervalos de tempo.

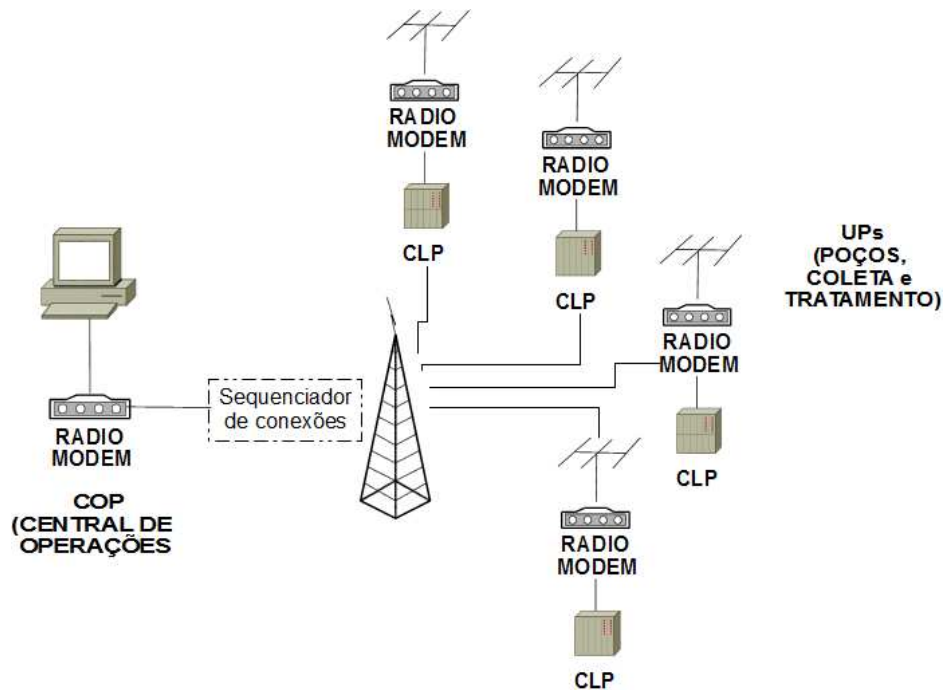



Figura 1: Arquitetura de automação do campo

No modelo ilustrado na Figura 1, as decisões envolvendo o conhecimento do especialista são feitas na COP e repassadas aos poços, através de um sistema que permite ao operador selecionar uma dentre algumas ações disponíveis. Quando necessária a intervenção direta no poço, os operadores de campo são contatados para que uma visita *in loco* seja feita.

Alternativamente a arquitetura cliente-servidor para sistemas distribuídos, as arquiteturas de pares denominadas *Peer-to-Peer* apresentam uma forma de distribuir parte da tarefa computacional para as unidades, e assim, melhorar a eficiência e aumentar a tolerância a falhas do sistema, uma vez que a dependência de um ponto central é diminuída.

A utilização de uma arquitetura descentralizada permitiria, entre outras coisas, enxergar o campo como uma sociedade de indivíduos interrelacionados e cooperativos, melhorando a forma como os sistemas computacionais são aplicados.

Mas essa alternativa requer o uso de ferramentas computacionais, sejam elas software ou hardware, diferentes das normalmente utilizadas nos modelos centralizados de operação.

Por outro lado, nas últimas décadas a comunidade científica tem visto crescer um importante paradigma da computação, o dos sistemas baseados em agentes.

Agentes são pequenos subsistemas que ao serem colocados em um ambiente dinâmico

apresentam um comportamento autônomo e inteligente, ou seja, são capazes de agir com base em suas próprias percepções do ambiente.

A evolução no uso de agentes na resolução distribuída de problemas, onde componentes computacionais estão distribuídos fisicamente e interligados por meio de redes de comunicação, levou à denominação dos sistemas multiagentes (SMAs). Um SMA é composto de agentes com capacidades, tarefas e percepções próprias do ambiente do qual são encarregados, mas que seguem metas globais para a melhoria do sistema como um todo.

Os SMAs oferecem suporte para a construção de sistemas distribuídos, sobretudo, os baseados na arquitetura *Peer-to-Peer*. Eles têm sido utilizados em diversos campos, a exemplo robótica, sistemas de aprendizagem, supervisão e controle de sistemas de manufatura.

Jennings (2001) observa que, em um SMA, a inteligência[1] não necessariamente está em cada indivíduo (agente), mas emerge a partir da cooperação entre eles. Como exemplo estão as comunidades de formigas onde indivíduos com tarefas e capacidades simples formam uma sociedade complexa.

No que tange aos campo de produção de petróleo, novas técnicas passaram a ser empregadas, recentemente, com o objetivo de aperfeiçoar a operação de poços de petróleo. Entre elas, está a utilização da completação inteligente, alcunhada por termos como "poço esperto" ou "campo inteligente".

Um *poço esperto* é definido por Gao (2007) como "Poço equipado com equipamentos de medição permanente ou válvulas de controle do fundo do poço, e especialmente aqueles com ambos."

Segundo Steem (2006), um campo inteligente é um campo completamente integrado, remotamente controlado e operado, onde dados de pressão regular, temperatura, fluido e movimentação de fluido são continuamente obtidos e imediatamente transmitidos para o usuário final para monitoramento e controle em tempo real.

A evolução do conceito de campos inteligentes caminha na direção de alcançar de fato o gerenciamento integrado da produção e reservatórios ante condições de mercado, o que permitiria, por exemplo, que as operações de produção de um campo sejam determinadas dinamicamente em função da oferta e demanda de petróleo no mercado (SILVA, 2006).

Apesar dessa visão futura, poucas pesquisas têm direcionado a atenção em quais estruturas

---

1 O termo inteligência é aqui definido como a capacidade de um indivíduo ou grupo de entender e compreender as condições do ambiente e agir sobre ele.

ou tecnologias computacionais podem ser utilizadas para lidar com esse modelo de campo. Muitos casos de utilização do conceito de campo inteligente são feitos com estratégias de controle centralizado, que se limitam a entregar aos engenheiros do campo os dados coletados em tempo real, como as experiências divulgadas em Al-Arnaout, Al-Driweesh e Al-Zarani (2008).

Este trabalho pretende, assim, estudar a organização dos poços em um campo de produção de petróleo, com o objetivo de projetar um sistema baseado em SMA que permita a realização de gerenciamento inteligente a nível local, ou seja, implantado junto a UP, visando a melhoria de desempenho do campo de produção como um todo.

Esse sistema permitiria, ainda, levar à UP o conhecimento especialista para o diagnóstico das condições operacionais do mesmo, distribuindo o controle, e assim aumentando a tolerância a falhas do sistema, além de oferecer um melhor aproveitamento dos recursos computacionais. A utilização de SMA permite ainda a cooperação e coordenação entre as UPs conferindo ao sistema a capacidade de resolução de problemas que afetam o desempenho geral do campo.

Esta proposta visa ainda uma distribuição do sistema em partes menores, com tarefas específicas (agentes), de forma a permitir a sua implantação em dispositivos de hardware com recursos limitados.

1.1 OBJETIVOS DA PESQUISA

Este trabalho tem por objetivo principal construir um sistema baseado em sistemas multiagentes para o gerenciamento integrado de campos de produção de petróleo de forma distribuída, com alto nível de autonomia e com baixo custo. Tal objetivo foi subdividido em cinco objetivos secundários, que são:

- Definição dos requisitos do sistema pelo estudo das técnicas de gerência de campos de produção e dos conceitos de campo inteligente;
- Modelar um sistema de gerenciamento autônomo utilizando paradigma de aplicações multiagentes;
- Desenvolver um sistema de gerenciamento autônomo e embarcado utilizando um modelo de aplicações multiagentes;
- Implantar o sistema em um hardware que possibilite a execução local de análise, diagnóstico de condições, e registro de informações coletadas em campo;

- Integrar a solução proposta em um ambiente simulado que permita sua validação.

1.2 JUSTIFICATIVA

Este trabalho propõe a utilização da abordagem baseada em agentes inteligentes e sistemas multiagentes para a construção de um sistema de gerenciamento de campo de produção de petróleo.

A utilização de SMA permitirá a execução local de tarefas que antes eram exclusivamente realizadas pelo operador na COP, isso por conta da organização proposta por SMA, que visa construir unidades de software autônomas, porém, com capacidade de interação com outras unidades, visando à solução distribuída de problemas.

A arquitetura proposta para SMA permite sua implantação em dispositivos com recursos limitados de processamento e armazenamento. Tais dispositivos podem ser representados por computadores de pequeno porte, mas que suportem as restrições operacionais impostas pelo ambiente.

Eles por sua vez podem ser instalados junto às UPs auxiliando os dispositivos de controle em tarefas que requerem um maior esforço computacional ou ainda que dependam de informações externas.

A colaboração entre as unidades de produção permite, por sua vez, que informações obtidas por outras unidades sejam utilizadas localmente para inferir diagnósticos que, apenas analisando suas condições locais, não seriam possíveis.

1.3 METODOLOGIA

Inicialmente foi realizada uma revisão nas referências técnicas e bibliográficas sobre campos de petróleo observando-se os aspectos ligados a sua organização, operação e manutenção, com o objetivo de identificar situações onde as unidades sofrem intervenções por parte do operador, com o objetivo de manter o sistema operando dentro do plano estabelecido para o campo. Paralelamente, foi realizada uma pesquisa sobre os modelos de sistemas multiagentes obtendo um conhecimento teórico para balizar a construção do sistema proposto.

Após a pesquisa teórica foi realizada a implementação do sistema, o qual foi testado de maneira experimental no Laboratório de Elevação Artificial da UFBA, utilizando um Sistema de Bombeio em escala reduzida, integrada a componentes simulados das outras etapas da produção (coleta e tratamento). A seguir um resumo dos passos realizados:

-Revisão do tema “campos de produção de petróleo e campos inteligentes”;

-Revisão do tema “sistemas multiagentes”;

-Estudo das tecnologias para o desenvolvimento de sistemas multiagentes;

-Modelagem conceitual de um campo de produção de petróleo;

-Identificação dos requisitos para a gerência de um campo de petróleo;

-Especificação de um sistema multiagente para a gerência do campo;

-Implementação do sistema;

-Realização de simulações e testes;

-Avaliação dos resultados.

1.4 ESTRUTURA DA DISSERTAÇÃO

O trabalho está organizado em seis capítulos.

O capítulo 2 aborda alguns conceitos que formam a base teórica utilizada para fundamentar algumas escolhas feitas nesta pesquisa. São eles: desenvolvimento de sistemas baseado em agentes, paradigma no qual esse trabalho está fundamentado; sistemas distribuídos, seus principais desafios e opções de arquiteturas; sistemas de produção de petróleo que constituem o cenário para o desenvolvimento e validação da pesquisa; e, por fim, apresentação de um modelo para a construção de sistemas de controle industriais distribuídos, que serviu de referência para algumas estruturas usadas na solução proposta.

O capítulo 3 traz uma revisão sobre o uso do paradigma de agentes na construção de sistemas de controle industrial, suas motivações e alguns casos de utilização do paradigma em sistema de manufatura.

O capítulo 4 apresenta o sistema proposto na pesquisa. São demonstradas inicialmente, suas principais motivações, bem como benefícios esperados com a utilização da tecnologia de SMA no

controle da operação de um campo de produção de petróleo. A construção do sistema está dividida em duas etapas: o projeto, onde é definida toda a estrutura conceitual do sistema, e a implementação, onde é demonstrado como o conceito foi transformado em artefatos de software.

O capítulo 5 apresenta a validação experimental do sistema construído. São apresentados, inicialmente, a estrutura laboratorial para a realização dos experimentos e os casos de testes conduzidos para validar o funcionamento do sistema.

O capítulo 6 traz as conclusões da pesquisa, contribuições, limitações e propostas de trabalhos que podem surgir a partir das idéias aqui expostas.

O trabalho inclui ainda seis anexos tratando: apresentação do ambiente para o desenvolvimento de SMA, representação do conhecimento, funcionamento básico de uma Unidade de Bombeio (UB), definições do modelo de referência, montagem das bases de conhecimento e documentação dos agentes.

2 FUNDAMENTOS

Este capítulo apresenta alguns conceitos que servirão como base para o entendimento do trabalho ou como fundamentação para alguns conceitos adotados no seu desenvolvimento. Inicialmente são apresentados o conceito de sistemas distribuídos e os desafios para sua construção, bem como as principais arquiteturas: Cliente-Servidor e *Peer-to-Peer*.

Em seguida é discutido o desenvolvimento de sistemas baseado em agentes, as definições para o termo "agente inteligente" e "sistemas multiagentes".

São apresentados também alguns fundamentos importantes sobre a indústria de produção de petróleo tais como métodos de elevação artificial utilizados e a estrutura dos campos de produção, além de uma visão geral da tecnologia de campo inteligente e como a comunidade científica o entende e define.

Por fim, é apresentado o resumo de um modelo para o desenvolvimento de sistemas de controle industrial autônomo e distribuído (GCAD) que serviu de base para a construção de alguns componentes do sistema apresentado.

2.1 SISTEMAS DISTRIBUÍDOS

Diversas definições para Sistemas Distribuídos (SDs) podem ser encontradas na literatura, o que demonstra a dificuldade em definir o que seria ou não um SD. Para Tanenbaum (2008), uma definição mais abrangente seria "um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente". Outra definição bastante aceita é dada por Colouris (2007): "Um sistema no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas enviando mensagens entre si".

Essas definições consideram que os SDs são formados por componentes (que são os computadores) que devem ser autônomos. Os usuários (que podem ser pessoas ou programas) precisam, por sua vez, manter a visão de estarem interagindo com um único componente, o que sugere, de alguma forma, a idéia de cooperação ou colaboração entre eles.

No entanto, essas afirmações não estabelecem nenhuma premissa quanto ao tipo de computadores utilizados, nem tampouco sobre como eles estão interligados. O propósito disso é não excluir sistemas formados por computadores distintos, desde computadores centrais até nós pequenos e limitados, como os que formam uma rede de sensores[2].

Segundo Colouris (2007), a busca por um melhor aproveitamento dos recursos, além de escalabilidade, flexibilidade e modularidade são motivações para a construção de SDs.

A escalabilidade sugere que um sistema possa suportar um crescimento escalar do número de usuários ou ainda do volume de dados processados pelo mesmo, enquanto que a flexibilidade consiste em produzir um sistema capaz de lidar com alterações em sua estrutura, como a adição de novas funcionalidades ou componentes (TANEMBAUM, 2008).

A modularidade consiste em dividir um sistema em subpartes, permitindo, assim melhor tratar problemas complexos.

2.1.1 Desafios e metas dos sistemas distribuídos

Tanembaum (2008) afirma que, apesar de existir tecnologia disponível para a implementação de SD, isso não quer dizer que essa sempre será a melhor alternativa. Isso porque, desenvolver SD significa lidar com desafios tais como: abertura, escalabilidade, heterogeneidade e transparência.

Desses aspectos, abertura e heterogeneidade são mais relevantes para essa pesquisa, e serão descritos a seguir.

i) Abertura

Um sistema distribuído aberto é aquele que segue normas e padrões para os serviços por ele

2 Uma Rede de Sensores é formada por diversos nós dispersos geograficamente, com o objetivo de monitorar algum fenômeno, os nós são interligados por meio de uma rede sem fio. Cada nó é formado por um sensor e dispositivos para o armazenamento e transmissão dos dados coletados. Eles são extremamente reduzidos não ultrapassando 0,1m de comprimento e largura.

disponibilizados, seja na descrição sintática ou semântica dos mesmos (TANEMBAUM, 2008). A abertura é um desafio importante a considerar na construção de um SD, uma vez que componentes irão interagir e cooperar.

Os protocolos de rede, como o TCP/IP, são exemplos dessa abertura. No caso dos sistemas, é comum a utilização das chamadas *Interface Definition Language* (IDL), com o intuito de descrever a sintaxe dos serviços oferecidos por um componente ou sistema.

O uso de tais definições permite que processos que utilizam serviços de outros sejam construídos sem se conhecer os detalhes da implementação, bastando, para isso, conhecer sua interface. E ainda que uma mesma interface possua implementações distintas para SDs distintos, mas apresentam funcionamento semelhante (TANEMBAUM, 2008). Exemplo disso seria uma interface que especifica um serviço de armazenamento de dados implementada para armazenamento em arquivos e em bancos de dados relacionais.

ii) Heterogeneidade

A heterogeneidade consiste em lidar com diferentes tipos de hardware ou plataformas de sistemas e atinge (COULOURIS, 2007):

- redes;
- hardware de computador;
- sistemas operacionais;
- linguagens de programação;
- implementações diferentes por diferentes desenvolvedores.

Dois importantes aspectos da heterogeneidade a serem considerados são: interoperabilidade e portabilidade (TANEMBAUM, 2008).

Interoperabilidade é a característica de um subsistema operar em conjunto com outros de diferentes plataformas, quer sejam de software ou hardware (TANEMBAUM, 2008).

Já a portabilidade consiste na característica que um sistema pode oferecer de ser transferido para outra plataforma diferente da qual foi inicialmente projetado sem a necessidade de alterações em seu código.

Lidar com a heterogeneidade é, atualmente, um dos maiores desafios ao se construir SDs, pois as redes de computadores tornaram-se cada vez mais acessíveis, permitindo dessa forma a utilização de dispositivos antes imaginados apenas na ficção, tais como: celulares, eletrodomésticos, veículos, máquinas industriais e robôs.

2.1.2 Arquiteturas de sistemas distribuídos

A arquitetura de um sistema busca expor, de forma clara, quais componentes o formam e como esses componentes podem se interligar e cooperar para prover seus serviços. A arquitetura de um SD tem implicação direta no desempenho, confiabilidade e segurança do sistema (COLOURIS, 2007).

A arquitetura pode ser compreendida como a divisão de responsabilidades dos componentes e sua localização nos computadores na rede. Cliente-servidor e *Peer-to-Peer* são exemplos de arquiteturas (COLOURIS, 2007).

a) Cliente-servidor

Na arquitetura cliente-servidor, um processo denominado “cliente” interage com outro denominado “servidor”, solicitando ao mesmo um serviço. Para isso, o cliente envia uma mensagem empacotando os dados necessários para atender sua solicitação e aguarda até que o servidor tenha processado o pedido e lhe envie uma resposta (TANENBAUM, 2008). Um servidor, por sua vez, pode atuar no sistema também como um cliente, quando, para a conclusão do seu serviço, o mesmo necessita recorrer a outro processo, a exemplo dos servidores de aplicações *web* que, frequentemente, para responder às interações feitas no navegador, precisa recorrer a informações de um servidor de dados responsável pelo armazenamento seguro das informações em um banco de dados.

b) *Peer-to-peer*

A arquitetura *peer-to-peer* é caracterizada por não existir a distinção entre processos cliente ou servidor. Nela, os processos comportam-se como pares, ou seja, cada nó oferece os mesmos serviços, agindo coletivamente para a conclusão de tarefas (COLOURIS, 2007).

Assim como na arquitetura cliente-servidor, os processos se comunicam enviando mensagens uns aos outros, e não acessando diretamente suas páginas de dados ou endereços de memória.

As noções aqui apresentadas sobre os SDs ajudaram a formar o conceito base para a construção deste trabalho, uma vez que o ambiente de execução do sistema, no caso um campo de produção de petróleo é formado por diversos subsistemas (poços de produção, estações de armazenamento e tratamento de óleo), a maior parte deles apoiados por computador.

Além disso, os tipos de computadores ou outros dispositivos (redes, dispositivos atuadores e sensores) criam um ambiente altamente heterogêneo, além da necessidade de compartilhar informações a fim de analisar e controlar o funcionamento do campo como um todo.

2.2 DESENVOLVIMENTO DE SISTEMAS ORIENTADO A AGENTES

Ao longo dos últimos anos diversos paradigmas para o desenvolvimento de sistemas computacionais foram surgindo. Alguns simplesmente como evolução dos que os precederam, outros com perspectivas completamente novas para projetar e implementar sistemas computacionais. Exemplos são os paradigmas estruturado e o orientado a objetos.

O paradigma orientado a objetos (POO) é, de fato, o mais difundido entre os desenvolvedores atualmente, perceptível pelo grande número de linguagens hoje utilizadas no mercado de desenvolvimento de sistemas, a exemplo de Java, C++, C#, VB.NET.

Apesar de o POO ser hoje bem aceito e largamente utilizado, ele não supre todas as necessidades, quando se trata de alta complexidade, como os industriais, que requerem softwares flexíveis, adaptativos e robustos (SILVA, 2003).

A programação orientada a agentes (POA), tem sido amplamente explorada pelos pesquisadores nos últimos anos, sobretudo para construção de sistemas complexos (SILVA, 2003). De fato, apesar de suas primeiras citações terem surgido ainda na década de 1970, este vem ganhando cada vez mais a atenção, quando o assunto é a construção de sistemas complexos.

2.2.1 Agentes

Um agente é uma entidade de software autônoma, também chamado de agente inteligente. O

termo "inteligente" é usado por entidade de software cuja seleção de suas ações é baseada em conhecimento (TVEIT, 2001).

Maturana et al. (2004) estende a definição anterior acrescentando ao agente a capacidade de interação com o ambiente, ao afirmar que "Um agente inteligente é uma unidade autônoma que encapsula o conhecimento de aplicações e é capaz de interagir com seu meio ambiente de forma inteligente."

A definição de ambiente depende da esfera de atuação do agente, podendo ser um computador, um processo organizacional, um software ou mesmo uma planta industrial.

Para Tweedale (2007), a teoria de agentes é resultado da convergência de diversas técnicas empregadas na ciência da computação, entre elas pode-se destacar a programação orientada a objetos, a computação distribuída e a inteligência artificial.

Para Girardi (2004), a autonomia é a principal característica do agente, isso porque é capaz de atuar sem intervenção humana, uma vez que realiza suas tarefas baseado na sua representação do ambiente.

2.2.2 Tipos e características dos agentes

Diversas características podem ser utilizadas para definir os agentes. Entretanto, antes, é preciso destacar que existem dois tipos de agentes: os reativos e os cognitivos. Agentes reativos (Figura 2) têm um comportamento baseado unicamente em suas percepções do ambiente, por isso apresentam geralmente um comportamento simples, e sua inteligência surge da cooperação entre os diversos indivíduos envolvidos na solução de determinada tarefa (HUBNER e SICHMAN, 2003).

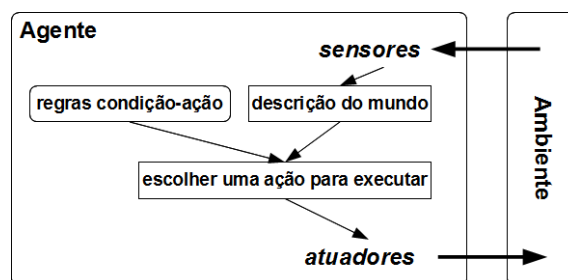



Figura 2: Modelo de agente reativo.

Já os agentes cognitivos (Figura 3), buscam cumprir seus objetivos elaborando um plano de ação a partir do seu raciocínio.

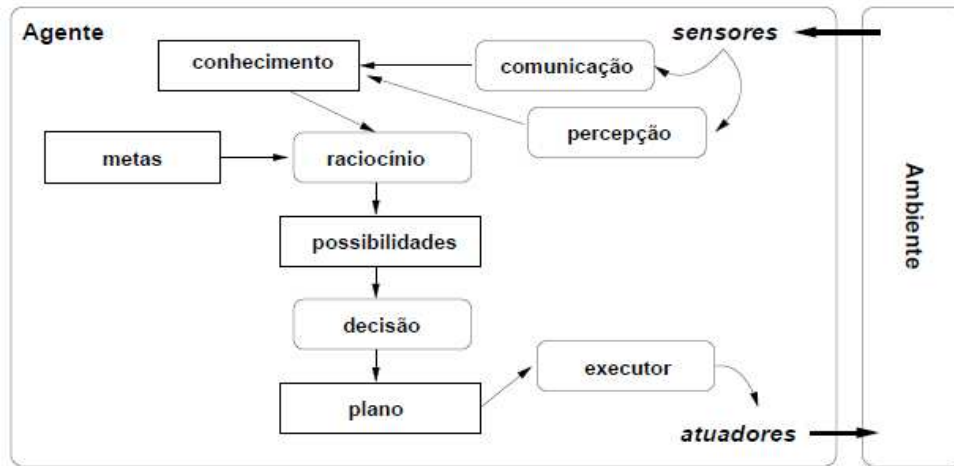



Figura 3: Modelo de agente cognitivo
Fonte: Demazeau (1990) apud Hubner (2003)

Segundo Bordini (2003), as características comumente utilizadas para descrever os agentes são:

- O agente é capaz de perceber alterações no ambiente;
- O agente age sobre o ambiente visando atingir o seu objetivo, e essas ações geram mudanças no ambiente;
- Comunicação é uma atividade essencial para assegurar a realização de tarefas globais que dependem da participação de diversos agentes;
- Possui uma representação simbólica do ambiente e sobre os demais agentes que compartilham com ele esse mesmo ambiente;
- Possui uma motivação que é representada pelos desejos e objetivos do ambiente, ou seja, qual o estado do ambiente que ele pretende alcançar;
- A partir de uma representação do estado atual do ambiente e uma motivação, o agente é capaz de decidir quais os estados futuros satisfazem o seu objetivo;
- Utiliza técnicas que permitem aprender sobre as percepções, crenças e motivação.

É importante destacar que os agentes reativos, normalmente, apresentam apenas as características de percepção, ação e comunicação, e, nem todo agente cognitivo precisa possuir todas essas características.

2.2.3 Agentes *versus* Objetos

Segundo Tveit (2001), a POA pode ser vista como uma extensão do POO, uma vez que no POO o objeto é o principal elemento e é caracterizado como uma estrutura lógica de programa, que engloba dados e funções (ou métodos) em um mesmo ponto. Como os objetos, na POA o agente é a entidade principal, entretanto, eles possuem, além das estruturas estáticas apresentadas pelos objetos, mecanismos para a representação de conhecimento sobre o ambiente que atua.

Os objetos interagem entre si por meio dos métodos, portanto, para se comunicar um objeto precisa conhecer a estrutura dos métodos do outro (SILVA, 2003). A Figura 4 ilustra uma comparação entre os agentes e objetos, onde os métodos, nos objetos, representam as interfaces com o meio externo, enquanto nos agentes esta vem dos “atos de comunicação” (FREITAS e BITTENCOURT, 2002), e suas ações não estão externamente visíveis como os métodos.

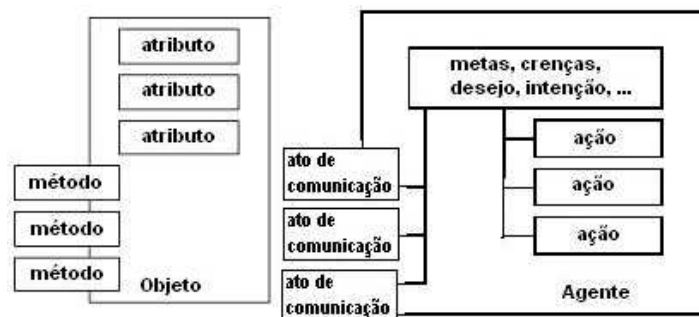



Figura 4: Comparação entre objetos e agentes
Fonte: Silva, 2003

Segundo Girardi (2004), uma característica que também distingue os agentes dos objetos é a autonomia, que pode ser observada pela sua capacidade de ativar os seus comportamentos, através do conhecimento embutido, das suas percepções do ambiente ou interações com outros agentes.

2.2.4 Sistemas multiagentes

Além de interagir com o seu ambiente, os agentes podem interagir com outros agentes cooperando entre si para a realização de tarefas complexas. Esse modelo é denominado sistemas multiagentes (SYCARA, 1998).

Um SMA pode ser definido como uma rede fracamente acoplada de indivíduos solucionadores de problemas, que interagem com outros solucionadores a fim de resolver problemas que estão além das suas capacidades individuais ou do seu conhecimento do problema (SYCARA, 1998; MONOSTORI, 2006).

Segundo Sycara (1998), as principais características que definem os sistemas multiagentes são:

a) Cada agente tem uma visão limitada ou incompleta do problema;

b) Não existe um sistema centralizado de controle;

c) As informações são descentralizadas; e

d) A computação é assíncrona.

Pode-se dizer que o paradigma de sistemas multiagentes surgiu da observação dos sistemas naturais, onde é possível perceber que o comportamento inteligente nem sempre parte das habilidades de raciocínio de um determinado indivíduo, mas pode ser alcançada pela interação com outros indivíduos (JOHNSON, 2001 apud HUBNER, 2004).

Exemplos disso podem ser percebidos nas estruturas simples que formam os neurônios e o complexo e poderoso mecanismo de inteligência formado pela união deles. Os formigueiros são outro exemplo dessa inteligência que emerge da coletividade (JOHNSON, 2001 apud HUBNER, 2004).

2.3 SISTEMAS DE PRODUÇÃO DE PETRÓLEO

A produção de petróleo compreende todo o conjunto de atividades necessárias para extrair o petróleo do reservatório e disponibilizá-lo para o transporte até as refinarias. Essa produção pode

ser organizada nas seguintes atividades: Elevação, Coleta, Tratamento, Transferência, Movimentação, Compressão, Injeção e Armazenamento (Figura 5).

Na etapa de elevação, o petróleo é extraído das rochas sedimentares dos reservatórios pela

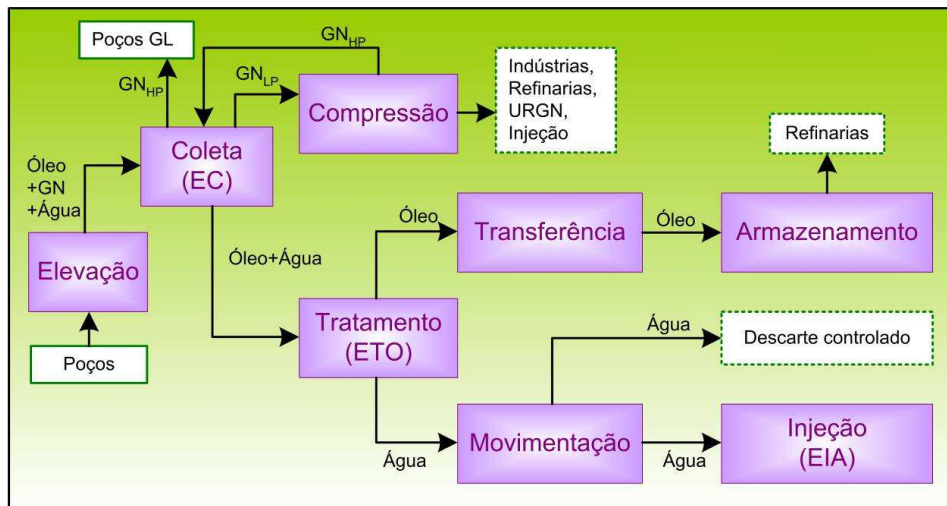



Figura 5: Visão geral de um sistema de produção de petróleo

coluna de produção até a superfície. Em seguida, o petróleo produzido é conduzido através de dutos às estações de coleta (EC), onde é separado segundo suas fases líquida (mistura de água e óleo), gasosa (GN) e sólida (sedimentos). A fase líquida é bombeada para a estação de tratamento (ETO), enquanto a fase gasosa escoa para a estação de compressores.

Nas ETOs (etapa de tratamento), a fase líquida é tratada e separada em tanques de óleo e em tanques de água produzida. Em seguida (etapa de transferência), o óleo tratado é bombeado para os Parques de armazenamento da Produção (etapa de armazenamento), de onde é distribuído para as Unidades de Refino.

A água produzida (etapa de movimentação) é bombeada às Estações de Injeção de Água (EIAs), a partir das quais pode ser direcionada para poços injetores de água (etapa de injeção), de forma a aumentar a pressão de fundo nos reservatórios, ou para o descarte controlado (BARRETTO, 2008).

2.3.1 Elevação Artificial

O petróleo é extraído do reservatório através da coluna de produção do poço, graças ao diferencial de pressão, que provoca o deslocamento do fluido para áreas de menor pressão. A pressão dos reservatórios pode ser naturalmente maior que o peso da coluna hidrostática de óleo na coluna de produção, o que promove, nesse caso, a surgência do fluido à superfície (poços surgentes). Caso contrário, a elevação deve ser apoiada por técnicas de elevação artificiais (BARRETTO, 2008).

Dentre as técnicas de elevação artificial mais utilizadas no Brasil, destacam-se: bombeio mecânico (BM), *Gás-Lift*, bombeio por cavidades progressivas e bombeio centrífugo submerso, sendo BM ainda a mais usada devido à sua versatilidade e confiabilidade, principalmente em se tratando de poços maduros e com baixa produtividade (THOMAS, 2004; PATRÍCIO, 1996).

O BM caracteriza-se, ainda, pelo baixo custo com investimentos e manutenção, flexibilidade na vazão e profundidade, boa eficiência energética e possibilidade de operar com fluidos com diferentes composições e viscosidades em uma larga faixa de temperatura (ORDONEZ, 2008). O Bombeio Mecânico (BM) é um sistema de elevação artificial composto por três partes: (i) a unidade de bombeio (UB) ou equipamento de superfície; (ii) a coluna de hastes; (iii) o conjunto de bomba de fundo.

Uma UB (Figura 6) é responsável pelo fornecimento de energia à bomba. Os componentes de uma UB típica são: motor, redutor de velocidade, unidade de bombeio, haste polida e cabeça de poço. O detalhamento do funcionamento do SBM é visto no Apêndice C.

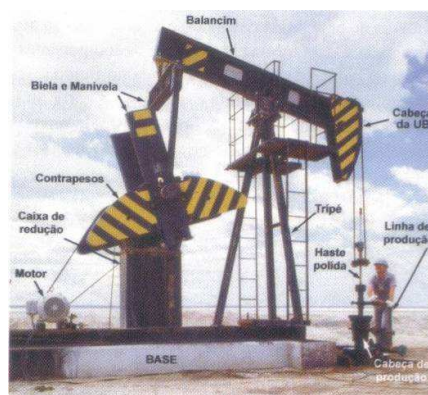



Figura 6: Unidade de Bombeio
Fonte:: Thomas, 2004

No presente trabalho, o SBM foi utilizado como método de elevação a ser controlado na etapa de extração. Ele foi utilizado por apresentar um número representativo de condições que podem ser simuladas em laboratório, além disso, pela existência de um SBM completo e com dimensões reais disponível para a realização dos experimentos de validação do sistema proposto.

2.3.2 Estações de Coleta (ECO)

As ECOs, representadas pela Figura 7, são responsáveis pela coleta do petróleo de um ou mais campos produtores, e pela separação e distribuição do fluido nas fases gasosa (GN) e líquida (óleo e água produzidos).

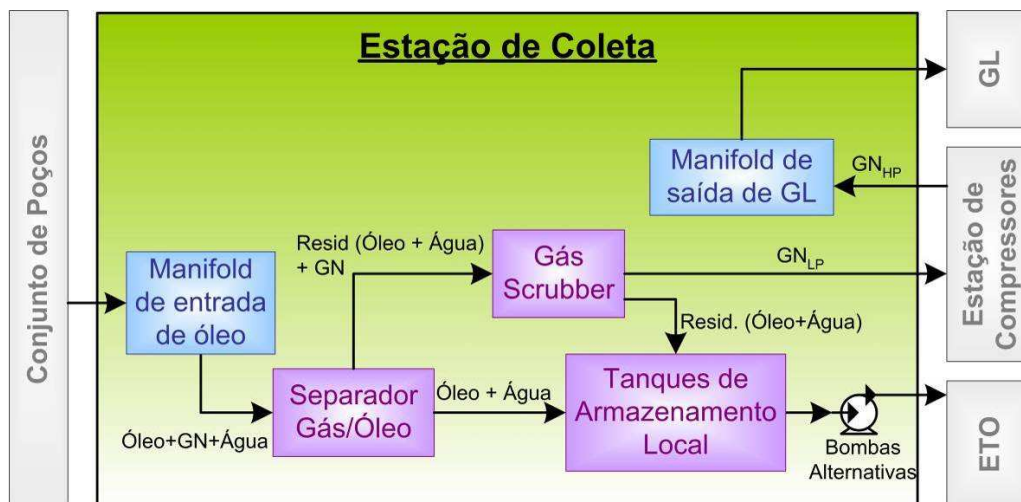



Figura 7: Visão geral de uma estação de coleta simples

Uma ECO é composta por *manifolds* de entrada de óleo, vasos separadores (separador gás/óleo e gás *scrubber* – separador bifásico utilizado para separação fina de gás), tanques de armazenamento local, dutos por onde o óleo é escoado, *manifolds* de saída de GN, para poços GL, e um conjunto relevante de bombas de transferência, que são utilizadas para realizar o transporte do fluido pelos dutos para outras estações. Além disso, contém dispositivos auxiliares e de monitoramento, como medidores de vazão, pressão e nível.

Algumas ECO possuem também equipamentos para realizar tratamento do óleo antes de bombeá-lo para a estação seguinte, de forma a minimizar o percentual de água presente no óleo, de acordo com os padrões de qualidade definidos pelas refinarias (ARAUJO, 2007; BARRETTO, 2008).

A ECO será utilizada na validação do sistema proposto. O sistema contará com acionamento automático das bombas que transportam o fluido para as outras etapas, a fim de manter o nível da ECO nos padrões desejados.

2.3.3 Estações de Tratamento de Óleo (ETO)

Uma ETO (representada na Figura 8) realiza o tratamento do fluido coletado, separando óleo e água, e armazena localmente as duas fases do fluido em tanques de óleo e de água produzida, respectivamente.

Figura 8: Visão geral de uma estação de tratamento

Uma ETO é composta pelos seguintes elementos principais: tanques de chegada da mistura óleo+água, tratadores de óleo, tanques de óleo, tanques de água produzida e bombas alternativas, para transferência às EIAs e aos Parques de Armazenamento. Nos tratadores de óleo, podem ser injetados tensoativos para auxiliar na separação da mistura óleo+água, através do aumento da tensão superficial da mistura.

A ETO também será utilizada para validação do sistema, uma vez que, junto a ECO e ao poço formam as três principais etapas da produção. As ações sobre essa unidade da produção, da mesma forma que na ECO, serão aplicadas no acionamento das bombas que transportam o fluido para as demais etapas. Dessa forma, o sistema visa a demonstrar como a produção final do campo pode ser gerenciada de maneira integrada e autônoma.

2.3.4 Gerenciamento de campos de produção

O gerenciamento de um campo de produção consiste em analisar variáveis de produção dos poços, com o objetivo de definir, individualmente, estratégias de produção para os poços que permitam maximizar a produção do campo (ALMEIDA, 2007).

É também uma tarefa que envolve diversas habilidades e disciplinas. Isso porque dados de diversas naturezas devem ser observados, tais como: aspectos físicos, operacionais e econômicos. A união dessas habilidades resulta em uma tarefa complexa e dispendiosa (SAPUTELLI, NICOLAOU e ECONOMIDES, 2005).

Outra atividade do gerenciamento de campos de produção é controlar a operação dos diversos poços, objetivando maximizar a produção (ALMEIDA, 2007).

Nas tarefas de gerência, o operador realiza atividades de intervenção nos poços, como: isolamento de intervalos produtores, abertura de novos intervalos, acidificações, fraturamento e outras operações de restauração. Porém, tais atividades normalmente estão associadas a altos custos, tornando inviáveis algumas dessas operações (SILVA, 2006).

2.3.5 Campo inteligente

Os termos "completação inteligente", "poço esperto", e "campo inteligente" têm sido utilizados para descrever um conjunto de tecnologias usadas com o objetivo de melhorar o desempenho da operação dos campos de produção de petróleo.

A completação inteligente consiste em equipar um poço com equipamentos de medição permanente de fundo do poço ou válvulas de controle (STEEM, 2006).

Os termos "poço esperto" ou "poço inteligente" são usados para descrever os poços equipados com a completação inteligente.

O que se pode observar é que, apesar de alguns autores considerarem completação inteligente um sinônimo de poço esperto, as citações para o segundo vão além, ao descrever suas habilidades, como pode ser visto na definição dada por Gao (2006), segundo a qual, um poço esperto é capaz de coletar, transmitir e analisar dados de completação, produção e reservatório e permitir controle seletivo de zonas para otimizar o processo de produção, sem intervenções físicas. Para isso, são requeridos dispositivos de controle de fluxo, *packers* de isolamento, cabos de força,

controle e comunicação, sensores de fundo do poço, aquisição de dados e controle de superfície (GAO, 2006). Na Figura 9 é ilustrado um esquema de um poço esperto.

Já o termo "campo inteligente" é utilizado para definir um campo no qual a tecnologia de poço inteligente é empregada e, além disso, são utilizadas tecnologias de comunicação, de forma a permitir que informações de completação, produção e reservatório possam ser utilizadas na tomada de decisões próximas do tempo real (MARTINEZ, 2002).

Segundo Steem (2006), um campo inteligente é um campo completamente integrado, remotamente controlado e operado onde os dados de cada poço são continuamente obtidos e imediatamente transmitidos para o usuário final para monitoramento e controle em tempo real.

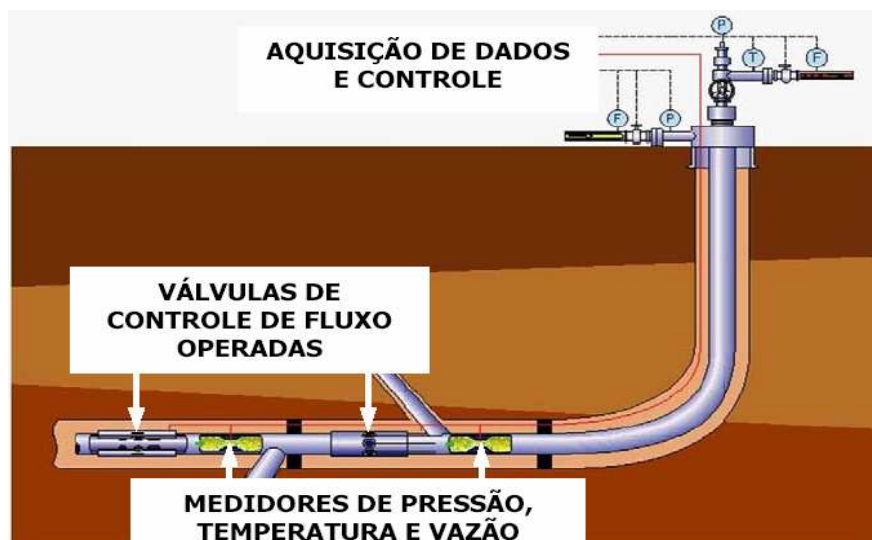



Figura 9: Esquema de um poço com completação inteligente
Fonte: Silva, 2006

Al-Arnaout, Al-Driweesh e Al-Zarani (2008) relata as experiências com a adoção da tecnologia de campo inteligente em Harad-III, que se baseou na implantação de uma rede de comunicação de fibra ótica que liga todos os poços para o sistema de supervisão e controle do campo (SCADA). Dessa forma, os dados adquiridos a partir dos poços são transmitidos imediatamente para o engenheiro responsável por controlar a operação do campo.

O que se pode observar nos relatos encontrados na literatura é que a tecnologia de campo inteligente, apesar de bastante empregada atualmente, tem-se limitado à utilização da tecnologia de poço esperto e à instalação de redes de comunicação, permitindo, dessa forma integrar as unidades da produção ao SCADA do campo.

Nas experiências relatadas, o benefício está em levar ao engenheiro responsável pela produção os dados imediatamente coletados dos poços, para que o mesmo disponha de ferramentas para melhorar o controle do campo.

Para o futuro do conceito de campos inteligentes, espera-se alcançar, de fato, o gerenciamento integrado da produção e reservatórios, o que permitiria, por exemplo, um campo em que as operações de produção seriam determinadas dinamicamente em função da oferta e demanda de petróleo no mercado (SILVA, 2006).

Entretanto, pouco é descrito sobre como as técnicas computacionais de hoje, como SDs e Inteligência Artificial, podem ser empregadas em conjunto com a tecnologia de campo inteligente, permitindo melhor explorar as vantagens de monitoramento e controle em tempo real de um poço esperto e o gerenciamento integrado dos campos inteligentes.

Sem pretensões de suprir essa lacuna, esse trabalho apresenta uma estratégia para o controle em nível local, mas com capacidade de comunicação e cooperação em nível de campo representando, assim, uma ferramenta que pode ajudar a aumentar o poder de análise e gerenciamento de um campo de produção de petróleo.

2.4 CONTROLE AUTÔNOMO E DISTRIBUÍDO PARA SISTEMAS INDUSTRIAIS AUTOMATIZADOS

O GCAD (Controle Autônomo e Distribuído para Sistemas Industriais Automatizados) é um modelo genérico para a construção de Sistemas Industriais Automatizados (SIA) geograficamente dispersos, proposto por Pacheco e Lepkson (2009). Nele, cada Célula Industrial que representa uma abstração de um Sistema Industrial Automatizado (SIA) local possui a habilidade de tomar decisões predominantemente baseada em sua Base de Conhecimento (BC). Porém, com a capacidade de compartilhar informações com outros SIAs aos quais esteja relacionado.

O GCAD foi utilizado como modelo de referência para a construção do sistema proposto, mais especificamente o Módulo Cognitivo (MC). As seções a seguir detalham a estrutura proposta.

Um resumo da proposta é ilustrado na Figura 10, onde são descritos quatro níveis principais de abstração: o nível gerencial, definido como Módulo de Gerenciamento Remoto (MGR), o nível cognitivo descrito pelo Módulo Cognitivo (MC), e o nível de dispositivo definido como Camada Reativa (CR) e o nível de processo representado pela Célula Industrial que são os dispositivos de campos tais como sensores e atuadores (PACHECO, 2011).

Cada célula autônoma é composta por MC e BC e está ligada aos dispositivos de campo através do MR.

O GCAD propõe a montagem do MC no próprio dispositivo de controle, conferindo-lhe a capacidade de comunicação horizontal, ou seja, com outros controladores em um mesmo nível da rede (PACHECO, 2011).

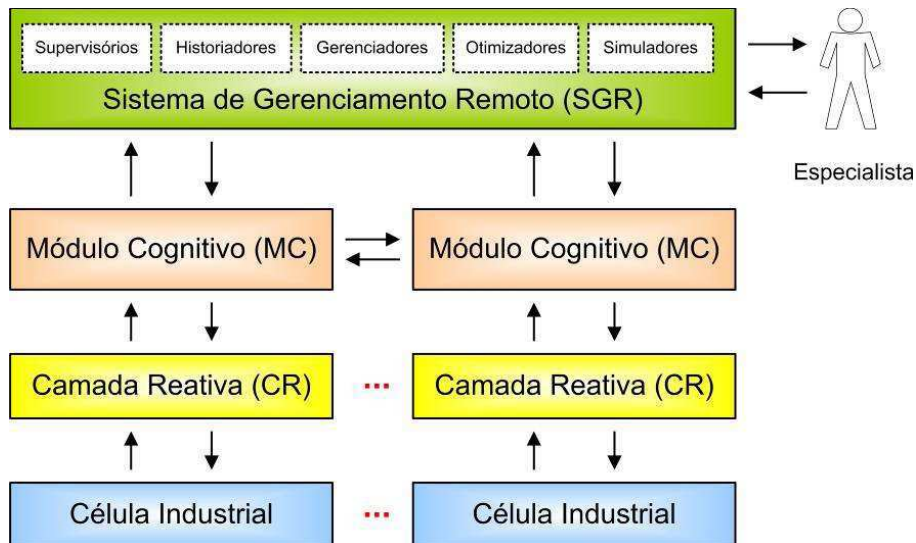



Figura 10: Modelo geral do GCAD
Fonte: Pacheco, 2011

Um MGR realiza o papel de supervisão geral de diversos SIAs, realizando tarefas como ajustes remotos, simulações e otimizações.

2.4.1 Módulo Cognitivo (MC)

Um MC é interligado a cada célula industrial e ao MGR, analisando, continuamente os dados do processo, visando identificar situações de anormalidade, as quais o controlador não pode responder, e propondo ajustes para correção. A Figura 11 ilustra a relação do MC com o MGR e outros MC.

O MC visa a exercer a função de um especialista na tomada de decisão, quando as situações envolvem informações imprecisas ou incompletas. Cada MC possui a capacidade de ler e escrever

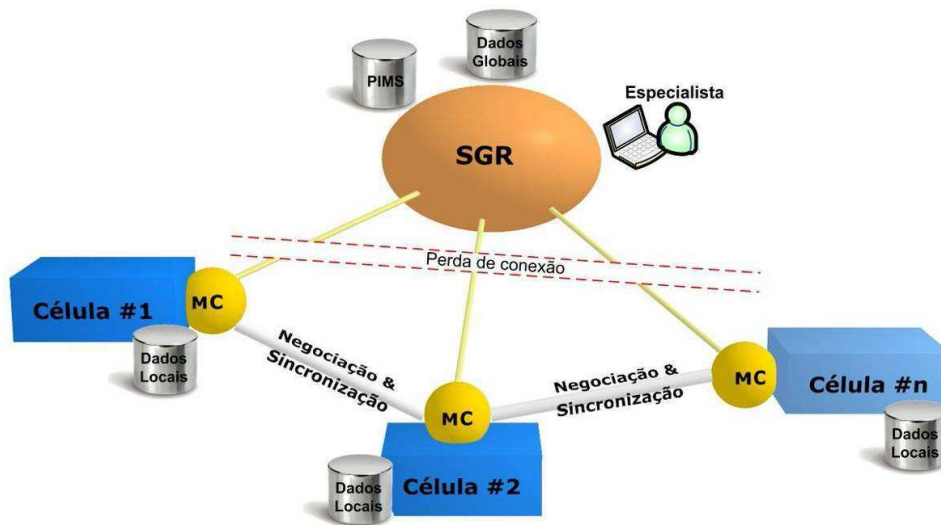



Figura 11: Relação do MC e MGR
Fonte: Pacheco, 2009

dados nos controladores, ajustando parâmetros ou *setpoints* (PACHECO, 2009). Ele traz ao GCAD dois importantes mecanismos para aumentar a eficiência dos diagnósticos e ações de correção locais, são eles, negociação e sincronização. O processo de negociação visa resolver possíveis conflitos causados por ações locais com reflexo em outras células. A sincronização permite disponibilizar dados de outras células, tais como, valores das variáveis e diagnósticos, que podem ser utilizados para aumentar a eficiência das decisões locais.

As operações de um MC são divididas em cinco módulos principais e dois complementares (Figura 12).

Figura 12: Módulos do MC
Fonte: Pacheco, 2011

São aqueles Verificação Continuada (MVC), Análise Local (MAL), Negociação (MNE), Sincronização (MSI), Atualização de Controle (MAC), e estes Tratamento de Exceção (MTE) e Gerenciador de Eventos (MGE).

i) MVC – tem como objetivo analisar os limites das variáveis obtidas do processo. As variáveis são analisadas quanto aos seus limites operacionais (limite de controle). Caso alguma variável ultrapasse o limite mínimo ou máximo preestabelecidos, o MCV deverá notificar ao módulo responsável por analisar essas variáveis.

ii) MAL – analisa as variáveis identificadas pelo MVC como fora dos limites de controle, tentando identificar ações corretivas cadastradas na base de conhecimento local, caso os diagnósticos não sejam conclusivos, ou seja, não levem à identificação de uma ação de correção, ele informa ao MSI para que proceda a propagação dos diagnósticos para outras células. Se uma ação corretiva for identificada, o mesmo verifica se sua execução pode afetar o funcionamento de outras células e, caso afirmativo, notifica o MNE para que proceda a negociação com as demais células envolvidas

iii) MNE – realiza a negociação com outras células, para o ajuste de variáveis locais que afetam o funcionamento de outras células.

iv) MAC – caso existam ajustes a serem realizados nos parâmetros do controlador e os mesmos tenham sido aprovados no processo de negociação quando necessário, o MAC procede ao ajuste, enviando ao controlador o novo valor para o parâmetro a ser alterado.

v) MSI – coordena a troca de informações entre as células, pode sincronizar valores das variáveis remotas requeridas pela célula local, permitindo a cada uma lidar de forma mais eficiente com problemas locais, dos quais não possui informações completas para estabelecer uma solução.

vi) MTE – é responsável por notificar ao MGR ou operador a ocorrência de alguma anormalidade que o MC não poderá solucionar.

vii) MGE – gerencia eventos externos à célula originados pelo MGR ou por CR.

No Apêndice D, estão as demais definições do GCAD, como, estrutura da BC e do MGR.

3 AGENTES NO DESENVOLVIMENTO DE SISTEMAS INDUSTRIAIS

A construção de Sistemas Distribuídos (SD), aplicados a sistemas industriais representa um dos maiores desafios para o desenvolvimento de sistemas hoje (SILVA, 2003).

Apesar da programação orientada a objetos (POO) ser hoje bem aceita e utilizada na construção de sistemas, ela parece não suprir todas as necessidades quando se trata de sistemas industriais, que requerem sistemas flexíveis, adaptativos e robustos (SILVA, 2003).

Por outro lado, a programação orientada a agentes (POA) tem sido amplamente explorada pelos pesquisadores nos últimos anos, sobretudo para construção de sistemas complexos (SILVA, 2003).

3.1 MOTIVAÇÕES PARA O USO DE SISTEMAS MULTIAGENTES

Durfee et al., (1989) apud Moulin e Chaib-Draa (1996), já observavam que muitos sistemas são naturalmente distribuídos, ainda que alguns sejam espacial e outros funcionalmente distribuídos. Portanto, o uso de SMA pode trazer diversas vantagens para o desenvolvimento de sistemas distribuídos, tais como:

- Resolução rápida de problemas - uma vez que SMA explora o paralelismo dos agentes, decisões podem ser tomadas enquanto ações estão sendo agendadas;
- Redução da comunicação - uma vez que apenas soluções parciais em alto nível são trocadas pelos agentes. Isso se deve ao fato do agente sempre buscar encontrar uma solução utilizando seu conhecimento e percepção locais;

- Maior flexibilidade - por possuir agentes com diferentes habilidades que podem ser dinamicamente relacionados para a resolução de um problema;
- Maior confiabilidade - por permitir que agentes assumam tarefas de outro agente em caso de falha.

É importante destacar que a redução na comunicação representa uma importante vantagem em relação aos sistemas que utilizam informações centralizadas e isso pode ser importante em sistemas de controle industriais, uma vez que dados brutos não são passados para outros nós do sistema, e sim conclusões ou observações construídas pelo agente local. (MOULIN e CHAIB-DRAA, 1996)

Por exemplo, ao invés de enviar conjuntos sucessivos de mensagens contendo os valores coletados a partir de um determinado sensor, para só então o destinatário concluir que aquela variável está fora do padrão desejado, um diagnóstico pode ser localmente realizado e apenas o resultado enviado para outro nó que, possivelmente, necessitará dessa informação, o que poderia resultar em uma única mensagem com a informação "variável X fora do limite de controle". Dessa forma também, a comunicação entre os nós deixa de possuir a necessidade de comunicação em tempo integral, podendo, inclusive, contar com estratégias de redes *ad hoc*[3].

Assim como a flexibilidade, que pode ser observada pelo fato novos agentes poderem ser acrescidos ao sistema para a realização de tarefas que antes não haviam sido designadas. Isso é possível devido ao fato de que, para serem autônomos, os agentes são construídos com o mínimo de acoplamento com outros.

Para isso, alguns ambientes para a construção de SMA possuem mecanismos para a descoberta de agentes e os serviços que eles oferecem. Dessa forma, um novo agente pode integrar-se a uma sociedade em funcionamento e conseguir cooperar, oferecendo seus serviços ou utilizando os serviços de outros (FREITAS e BITTENCOURT, 2002).

3.2 SISTEMAS MULTIAGENTES E OS SISTEMAS DE AUTOMAÇÃO

Segundo Wagner (2002), algumas propriedades intrínsecas aos sistemas de automação tornam interessantes a utilização de agentes em tais sistemas. Por exemplo, em sistemas que:

3 Redes *ad hoc* não possuem um nó central ou ponto de acesso, de forma que todos os nós da rede funcionam como roteadores. A informação é repassada de forma comunitária para cada nó vizinho.

- São complexos e distribuídos por natureza;
- Requerem diferentes visões ou habilidades. Engenheiros, gerentes e operadores tem uma visão individualizada da estrutura, dados e funcionalidades;
- Requerem softwares flexíveis e adaptativos;

De acordo com Jennings (2001), diversas ferramentas e métodos têm sido empregados para lidar com a complexidade desses sistemas, mas elas têm falhado entre outras, pelas seguintes razões:

1. Rigidez na interação entre os diversos subsistemas – em geral os sistemas são implementados em nível de campo por componentes de hardware, como os CLP. Nesses componentes a interação entre dois ou mais dispositivos é realizada através da leitura e escrita em suas tabelas de dados, de forma rígida, procedural e hierárquica;
2. Os mecanismos disponíveis são insuficientes para representar a estrutura organizacional dos sistemas – essas estruturas são normalmente definidas em tempo de projeto e não possibilitam mudanças em sua organização em tempo de execução.

Agentes podem ser agrupados em um subsistema, onde cada um é responsável por desempenhar uma tarefa e, para isso, interagem frequentemente. Além disso, os agentes de um determinado subsistema podem interagir com os de outros, conferindo maior capacidade para a resolução de problemas que afetam dois ou mais subsistemas (WAGNER, 2002).

Segundo Wagner (2002), dois importantes aspectos são particularmente significativos, quando se considera o uso de agentes em sistemas de automação. São eles:

1) Na abordagem baseada em agentes, a natureza descentralizada do problema é representada por múltiplos agentes, múltiplos locais de controle, múltiplas perspectivas e interesses conflitantes. Essa abstração leva a um melhor tratamento da complexidade desses sistemas e melhor reproduz as propriedades inerentes aos sistemas do mundo real.
2) Devido às interações de alto-nível, sistemas de agentes provêm mecanismos para a construção de estruturas organizacionais flexíveis, despertando para a possibilidade de construção de sistemas que se adaptam dinâmicamente às mudanças no ambiente.

3.3 EXEMPLOS DE APLICAÇÃO

SMAs têm sido utilizados em diversos campos de aplicação como, por exemplo, sistemas médicos, sistemas de aprendizagem e sistemas de busca por conteúdos em ambientes heterogêneos, como a internet. Outra área onde os agentes têm sido amplamente aplicados é na manufatura, particularmente em sistemas de automação industrial, a começar pela robótica, e nas tarefas de planejamento de processos, supervisão e controle de plantas ou processos industriais.

Essas tarefas requerem funções, tais como: observação do ambiente, gravação de informações, processamento de sinais e, a partir daí, diagnóstico de condições anormais, sobre as quais deve ser efetuado o controle. Podem ser observados sinais físicos, como vibração, deslocamento de material, temperatura, pressão, ou ainda, processos de negócio, tais como fluxos de trabalho, planos de produção (BUSSMANN, JENNINGS e WOOLDRIDGE, 2004).

Nas abordagens baseadas em multiagentes em manufatura, os subsistemas são identificados e classificados por tarefas como aquisição de dados, supervisão, controle e atuação, onde cada tarefa é dividida ou mapeada para um grupo de agentes responsáveis por executá-la e, ainda, estabelecer uma comunicação com os outros agentes, permitindo assim a execução de tarefas que contemplam o processo como um todo.

A seguir, são apresentados três casos da utilização de SMA em sistemas de manufatura. O primeiro refere-se ao planejamento da produção e alocação de recursos, e os demais, ao controle, diagnóstico e monitoramento de processos industriais. Eles foram selecionadas por estarem mais próximos do sistema de produção que servirá de base para a construção do sistema proposto.

3.3.1 Planejamento da produção e alocação de recursos – Caso 1

O processo de planejamento da produção consiste em selecionar e sequenciar um conjunto de tarefas com o objetivo de satisfazer as metas e restrições de domínio de uma organização (MONOSTORI, VÁNCZA e KUMARA, 2006).

Para Vrba, Hall e Maturana (2005) as abordagens centralizadas e hierárquicas, tradicionalmente utilizadas em planejamento e controle da produção ou gerenciamento da cadeia de suprimentos não podem lidar eficientemente com o alto grau de complexidade e a crescente necessidade de flexibilidade e robustez desses sistemas.

Por outro lado, as soluções baseadas em agentes vêm sendo vistas como alternativas para lidar com essa complexidade, sobretudo em atividades de reconfiguração de equipamentos onde o número de possibilidades pode apresentar um volume de alternativas impraticável (MONOSTORI, VÁNCZA e KUMARA, 2006).

Li *et al*. (2010) apresentam um sistema multiagente para o planejamento e alocação integrados de processos. A estrutura organizacional (Figura 13) do sistema é composta de 3 tipos de agentes: o agente tarefa, o agente máquina e o agente de otimização. Os planos de processos e a alocação desses planos para cada uma das tarefas são estabelecidos a partir da negociação entre os agentes.

Para cada tarefa um agente é instanciado e contém as informações particulares, como identificação, tipo, quantidades, requisitos de qualidade, projetos CAD (Computer-Aided *Design*), tolerância e requisitos finais. Para gerar as alternativas de planos de processo de cada tarefa, o agente utiliza uma base de conhecimento e negocia com os agentes de máquina gerando todas as alternativas de planos.

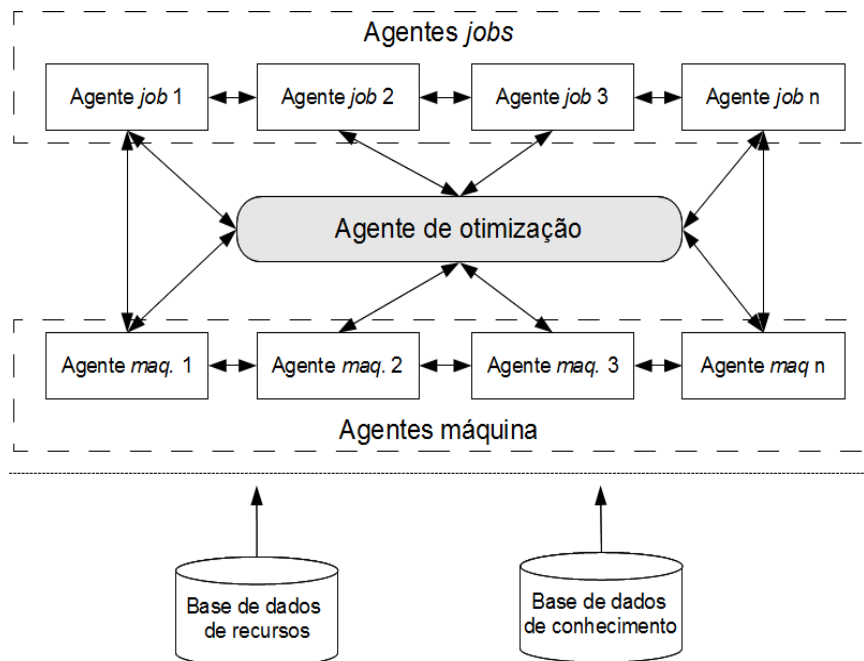



Figura 13: Organização dos agentes (traduzido de Li *et al.*, 2010)

O agente de cada máquina utiliza informações contidas em uma base de recursos com a identificação da máquina, a funcionalidade de fabricação que ela processa, o tempo de processamento e o estado dessa máquina. Os agentes de máquina negociam com os agentes de

tarefa para determinar que tarefas serão alocadas para cada uma e o tempo de processamento de cada tarefa.

O agente de otimização desempenha o papel de ajustar os planos e as alocações das tarefas para as máquinas. Ele utiliza, um algoritmo evolucionário, denominado *Modified* Genetic Algorithm.

Cada tipo de agente reside em um computador na rede, e a comunicação entre eles é feita através de uma plataforma de execução de agentes distribuídos. Ainda dois outros computadores mantêm as bases de recursos e conhecimento separadas.

Para os autores, a principal vantagem da utilização de SMA nas tarefas de planejamento de processos e alocação de recursos é que elas passaram a ser executadas simultaneamente, por meio da interação entre os agentes. Dessa forma, os recursos eram alocados, já pensando no melhor plano em função das condições das máquinas disponíveis.

Como o processo era conduzido de maneira autônoma pelos agentes, todas as possibilidades eram investigadas, porém levando um menor tempo para isso.

3.3.2 Monitoramento e diagnóstico de processo – Caso 2

Heck, Lenagle e Woern (1998) argumentam que, em uma instalação industrial moderna, existem componentes altamente complexos, conectados através de uma rede, e que ainda podem ser de diferentes fabricantes ou operar em diferentes plataformas.

Diversas estratégias de monitoramento e diagnóstico de sistemas industriais operam com arquiteturas centralizadas que, segundo Parunak (1994) apud Heck, Leangle e Woern (1998), não são efetivas como as abordagens baseadas em multiagentes distribuídos em rede.

Heck, Lenagle e Woern (1998) apresentam um sistema baseado em multiagentes para monitoramento e diagnóstico de sistemas industriais. O sistema consiste de três tipos de agentes, que são: o agente de monitoramento, o agente de diagnóstico e o agente facilitador, além de um quadro onde todos os diagnósticos são escritos, e ainda uma camada de comunicação responsável por isolar as diferentes plataformas utilizadas na implementação dos agentes. A Figura 14 ilustra a visão geral do sistema e como os agentes foram distribuídos nos componentes e dispositivos de diagnóstico.

O **agente de monitoramento** tem a finalidade de monitorar os componentes. Ele é responsável por enviar para o facilitador uma mensagem de erro, caso uma condição de falha ocorra em algum componente. A proposta é que esse possa ser implementado diretamente no controlador do componente, caso o mesmo resida em um PC, ou ainda implementado em linguagem Java e embarcado em um microprocessador local que possua capacidade de comunicação em rede.

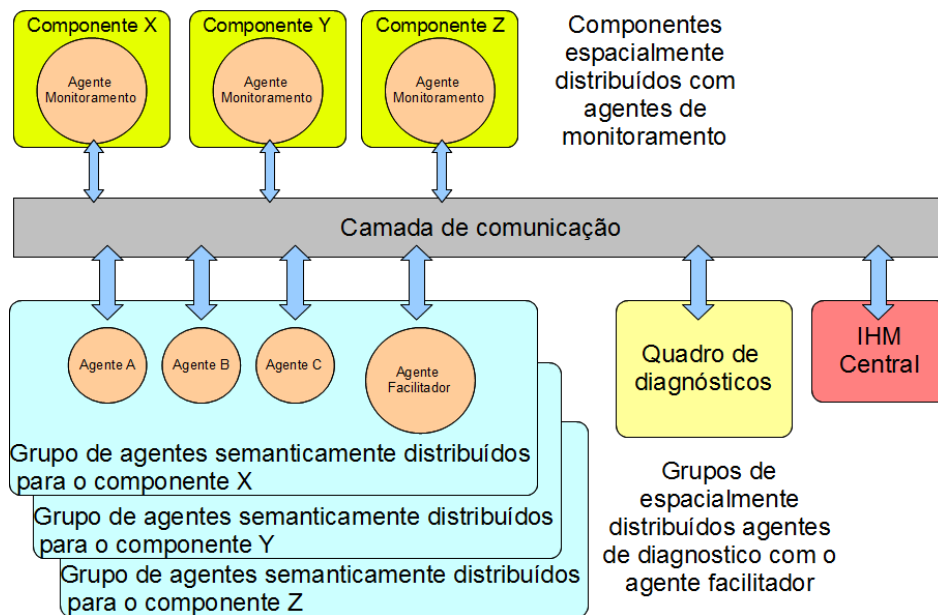



Figura 14: Visão geral do sistema (traduzido de Heck, Leangle e Woern, 1998)

O **agente de diagnóstico** é responsável por executar os métodos de diagnóstico para cada componente. Cada agente pode apresentar um método diferente de diagnóstico para um mesmo componente em uma estratégia que o autor denominou de distribuição semântica. Portanto, podem existir diversos agentes para diagnóstico com métodos e modelos apropriados para cada fabricante dos componentes de controle. Dessa forma, as regras para o diagnóstico de um componente podem ser encapsuladas no agente sem a necessidade de se conhecer as utilizadas por cada fabricante.

Por conta da distribuição semântica dos agentes de diagnóstico, o **agente facilitador** é responsável por resolver os conflitos, caso existam. Ele implementa, para tanto, dois mecanismos: o primeiro, para os conflitos semânticos, quando dois agentes de diagnóstico chegam a conclusões diferentes para um mesmo componente; o segundo, para a resolução de conflitos espaciais, quando componentes apresentam as mesmas condições de outros diagnosticados como faltosos, mas seu grupo de agentes de diagnóstico o identificou como em funcionamento normal.

O sistema possui ainda um módulo de interface com o usuário, onde os diagnósticos são apresentados ao operador.

3.3.3 Controle de processos – Caso 03

Vrba, Hall e Maturana (2005) apresentam suas experiências em utilizar o paradigma dos agentes para implementar sistemas de controle. O primeiro experimento foi conduzido em uma fábrica de barras de aço. O processo de fabricação das barras consiste em aquecer o aço e moldá-lo no tamanho desejado, utilizando, para isso, diversas esteiras e câmaras de resfriamento. Para a produção do aço, nem todas as esteiras e câmaras são utilizadas ao mesmo tempo, pois o sistema foi construído de forma a ser redundante e flexível. Portanto, para cada receita de aço, uma combinação de máquinas pode ser utilizada.

Após a análise dos principais requisitos para o sistema, um agente foi implementado para cada unidade participante (câmaras de resfriamento). O principal objetivo do SMA era agregar a capacidade de combinar quais câmaras poderiam ser utilizadas para atender aos requisitos de uma determinada receita. Um agente denominado *cooling box agent* foi implementado em linguagem "C". Executando em um PC (*Personal Computer*), sua tarefa era determinar a "saúde" da câmara e sua capacidade de resfriamento. Cada agente enviava uma "oferta" para um nó central, que arbitrava as ofertas para obter a correta curva de resfriamento, determinando quais câmaras seriam utilizadas no processo.

A segunda experiência foi a construção do *Real-Time Control Agent,* um SMA que integra as tarefas de tempo real dos controladores e a capacidade de decisão e cooperação dos agentes. Para essa integração foi utilizada uma abstração dos agentes denominada *holons* ou *holonics agents.* As tarefas de tempo real, como obter informações dos sensores e atuadores foram implementadas em uma linguagem de baixo nível para CLPs (Ladder). Já para as tarefas de alto nível, devido às necessidades de tomada de decisão e cooperação, os agentes foram implementados em uma linguagem de programação orientada a objetos.

Uma questão importante a ser observada nesse projeto é que os *holonics agents* foram implementados diretamente no CLP, conforme ilustra a Figura 15. Os agentes comunicavam-se com as rotinas de controle através da tabela de dados (*tags*) do controlador. Para a comunicação entre agentes, quer seja no mesmo CLP ou entre CLPs, foi utilizado padrão FIPA[4].

---

4 FIPA é um padrão para a troca de mensagens entre agentes de diferentes plataformas (FIPA, 2010).

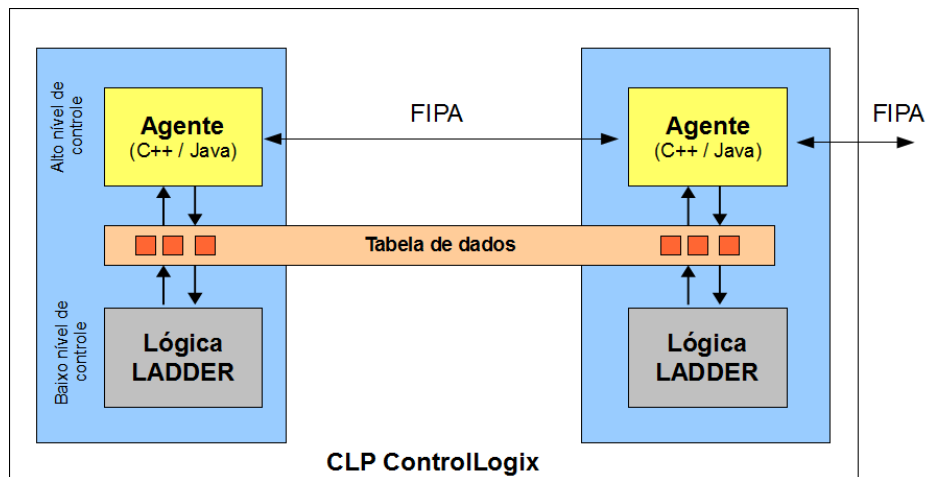



Figura 15: Arquitetura holonica (traduzida de Vrba, Hall e Maturana, 2005)

Para a codificação dos agentes após uma analise das principais ferramentas disponíveis, os autores decidiram por implementar uma solução partindo do zero. A plataforma ACS (Autonomous *Cooperative System*) foi implementada em C++ e foi desenvolvida buscando minimizar o uso de memória e processamento do CLP visando reduzir o impacto da solução nas tarefas de controle em tempo real do CLP.

Os autores argumentam que a utilização de agentes de software trouxe ao sistema maior flexibilidade, robustez e agilidade, uma vez que as interações dos agentes são mais dinâmicas, em contraste com as interações determinísticas das abordagens convencionais. Isso permite lidar mais facilmente com a reconfiguração dos equipamentos que formam o sistema de manufatura.

A utilização dos agentes permitiu também controlar o processo no nível de tomada de decisão, tarefas que, normalmente, cabem aos operadores. Essas decisões passaram, ainda, a envolver os diversos equipamentos que formam a planta. Como os agentes foram instalados diretamente nos controladores, as decisões poderiam ser tomadas mais rapidamente.

3.3.4 Considerações sobre os casos

As razões que levaram à escolha desses trabalhos foram, a similaridade com a proposta, ou utilização das idéias expostas no desenvolvimento do sistema proposto.

O caso 1, no entanto, não foi selecionado pela semelhança com a proposta ou por inspirar idéias, mas por representar um caso onde os agentes não monitoram ou controlam elementos físicos,

como máquinas ou computadores, eles atuam na observação de processos organizacionais e serve para demonstrar a diversidade de aplicações dos SMA no âmbito da manufatura.

A idéia de diagnóstico distribuído, apresentada no caso 2, inspirou um mecanismo semelhante, utilizado no sistema proposto, além de demonstrar como os agentes foram utilizados para lidar com a heterogeneidade dos componentes de hardware.

O caso 3 é o exemplo mais próximo do sistema de produção usado no trabalho. Além de demonstrar como SMA foi utilizado para lidar com a necessidade de visão global do funcionamento de um processo complexo, distribuído e flexível, ele traz a idéia de agentes serem embarcados no próprio CLP, o que combina com as propostas apresentadas neste trabalho, que é dos agentes atuarem próximos ao processo controlado, em uma estratégia diferente das alternativas de controle centralizado.

Além dos casos citados acima, diversos outros exemplos da utilização de SMA em sistemas de manufatura foram listados por, Monostori, Váncza e Kumara (2006) e Shen et al. (2006).

4 SISTEMA DE CONTROLE AUTONÔMO E EMBARCADO

Este capítulo apresenta o projeto do Sistema de Controle Autônomo e Embarcado para Campos de Petróleo Automatizados (SCAE), cuja motivação surgiu a partir da observação do modelo de operação de um campo de produção de petróleo.

A proposta do SCAE é fornecer uma infraestrutura computacional que permita descentralizar as tarefas de controle integrado de um campo de produção de petróleo, atuando de forma autônoma para controlar a sua operação.

Essa descentralização consiste em implantar, junto às UPs, aqui representadas por poços, estações de coleta e tratamento, um grupo de agentes com capacidade de obter os valores das variáveis a partir do controlador, inferir diagnósticos com base em uma representação do conhecimento especialista sobre o funcionamento da UP, identificar ações de ajuste nos parâmetros do controlador e comunicar-se com agentes residentes em outras UPs relacionadas.

A descentralização permite ainda manter certo nível de controle sobre a operação da UP, mesmo em caso de a central de operações inoperante por algum tempo, ou a UP perder a comunicação com as demais, isso porque diagnósticos continuam sendo identificados e ajustes repassados ao controlador local por meio dos seus agentes.

A comunicação com agentes de outras UPs leva em consideração a existência de um relacionamento entre as que formam um campo de produção. Essa comunicação proporciona um aumento significativo da capacidade de avaliação das condições operacionais locais, uma vez que tende a identificar situações de anormalidade que só poderiam ser feitas tendo uma visão das condições operacionais de outras UPs relacionadas. Um exemplo disso ocorreria no caso do rompimento em uma linha de distribuição que conduz o óleo extraído dos poços até as estações de coleta. Nesses casos, todos os poços envolvidos deveriam participar da decisão, bem como a ECO.

Espera-se com a utilização do SCAE reduzir o tempo tomado atualmente para a identificação de uma anormalidade no campo, e ainda permitir que ações de correção sejam automaticamente realizadas pelo sistema. Essa proposta é detalhada nas seções a seguir.

4.1 DEFINIÇÃO DE ESCOPO

A arquitetura geral do SCAE foi inspirada no GCAD, e é formada por um componente principal denominado Plataforma Local de Agentes (PLA), cujo correspondente no GCAD é o MC.

Cada UP do campo segue a abstração da Célula Industrial Autônoma (CIA) , que tem como requisito a comunicação com o MGR, aqui representado pela COP e demais UPs relacionadas. Na Tabela 1, estão relacionados os conceitos do SCAE e correspondentes do GCAD.

Tabela 1: Conceitos do SCAE x GCAD

SCAE	**GCAD**
UP	CIA
PLA	MC
COP	MGR

Na proposta do SCAE, cada UP recebe uma instância da PLA, que deve ser ligada ao controlador a fim de ler valores das variáveis locais e ainda ser capaz de alterar os seus parâmetros, ajustando o funcionamento da UP.

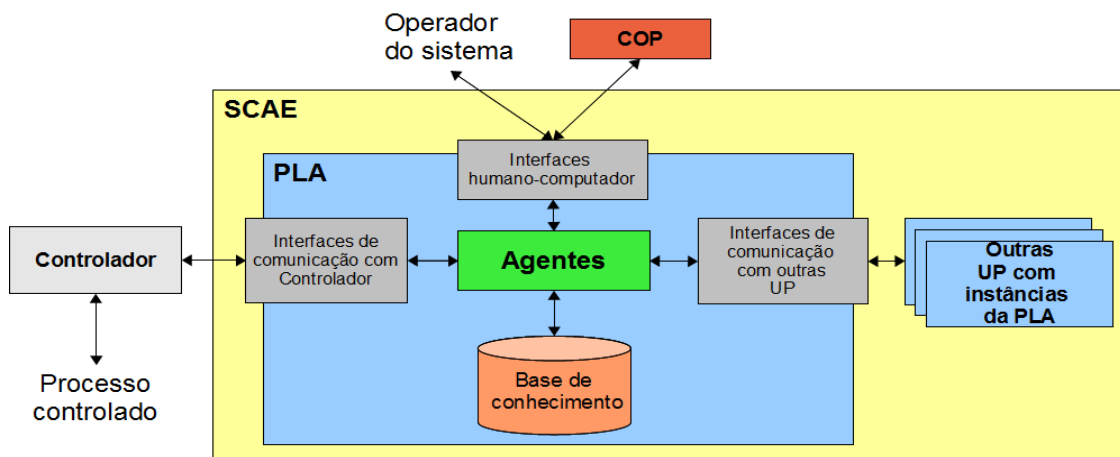



Figura 16: Visão geral do SCAE

A PLA é formada por agentes, base de conhecimento (BC) e interfaces de comunicação (Figura 16). Para cumprir sua função adequadamente, a PLA requer uma estrutura de hardware compatível com as necessidades requeridas de processamento, memória permanente e volátil, bem como comunicação.

4.1.1 Agentes

Os agentes são a espinha dorsal do SCAE, uma vez que toda a estrutura lógica da PLA é formada por agentes e seus comportamentos. Eles foram identificados visando prover ao SCAE habilidades necessárias para cumprir seu principal objetivo de atuar de maneira autônoma no controle da operação do campo.

A utilização dos agentes para construção do SCAE deve trazer ao campo os benefícios listados no capítulo 3 sobre a utilização de SMA em sistemas industriais. São eles: resolução distribuída de problemas, redução da necessidade de comunicação e maior flexibilidade.

A identificação e detalhamento dos agentes serão feitos mais adiante.

4.1.2 Base de conhecimento

A BC fornece aos agentes uma representação do ambiente (variáveis, UPs relacionadas, regras, dados operacionais, e configurações.) e é estruturada conforme Figura 17. Sua construção segue as definições da proposta apresentada no modelo GCAD e estão detalhadas no Apêndice D.

As regras representam o conhecimento dos especialistas acerca do funcionamento da UP. Elas são utilizadas pelos agentes para inferir diagnósticos, que são conclusões acerca de um conjunto de valores de variáveis em um dado instante e selecionar ações que representam quais ajustes podem ser realizados no controlador. A definição das regras é feita durante a fase de configuração inicial do SCAE para cada UP.

Os dados operacionais correspondem aos valores das variáveis adquiridas do controlador e são periodicamente requisitados pela PLA.

As especificações da UP fornecem a representação que permite ao agente “saber” quais variáveis devem ser requisitadas ao controlador, quais ações podem ser executadas e ainda quais outras UPs estão relacionadas.

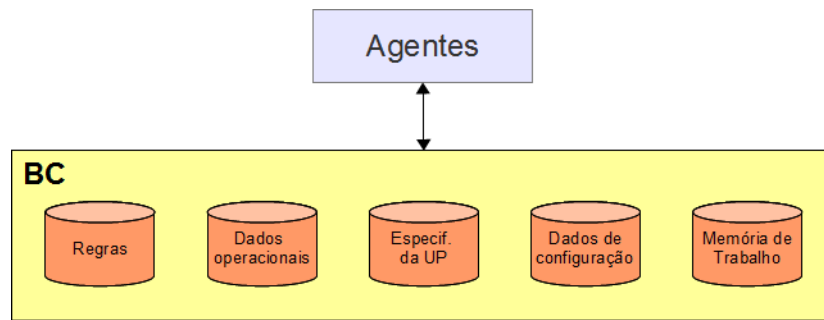



Figura 17: Estrutura da BC

Os dados de configuração são parâmetros para ajustar o funcionamento da PLA e, assim como as regras, são definidos na configuração inicial do SCAE.

A memória de trabalho fornece aos agentes uma visão de fatos presentes e passados. Nela são armazenados históricos de diagnósticos inferidos e de ações identificadas, além de valores atuais de variáveis. Uma definição completa da BC é dada em uma seção posterior.

4.1.3 Interfaces de comunicação

As interfaces do SCAE compreendem as formas de interação do mesmo com o ambiente ou meio externo, e são constituídas de (Figura 18):

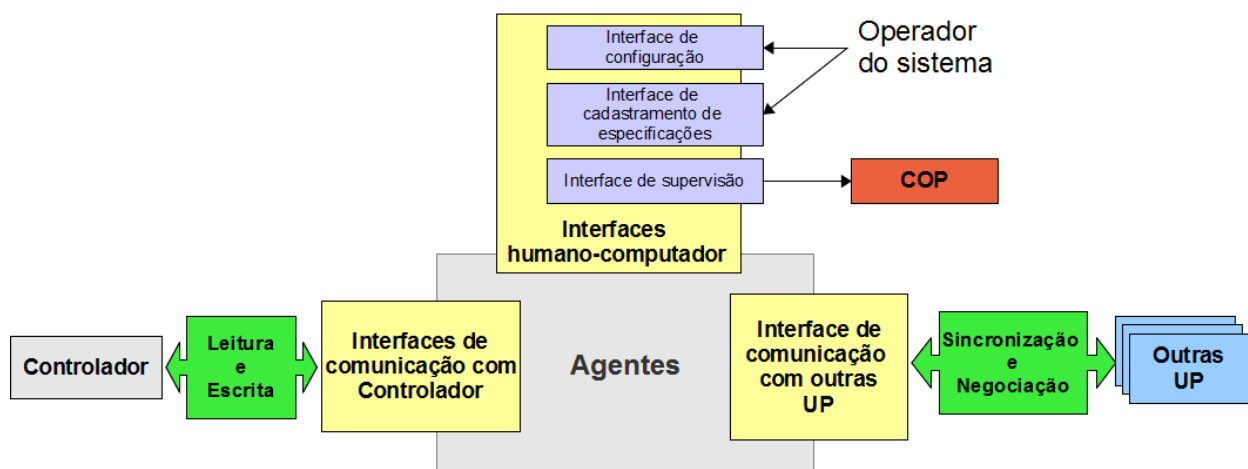



Figura 18: Definição das interfaces do SCAE

a) Interfaces de comunicação com o controlador – Permitem isolar do agente os detalhes de como é feita a leitura e escrita de dados neste. Dessa forma, quando um agente deseja comunicar-se com o controlador ele o faz por meio da interface correspondente.

b) Interfaces humano-computador (IHC) – Determinam como os usuários interagem com o SCAE, são elas:

   i) Interface de cadastramento das especificações e regras;

   ii) Interface de configuração;

   iii) Interface de supervisão.

c) Interfaces de comunicação entre UPs – Constituem um dos mecanismos mais importantes para o SCAE, pois elas especificam como os agentes de uma UP podem se comunicar com os que residem em outras UPs. Essa comunicação é feita com base nos mecanismos de sincronização e negociação definidos no GCAD.

   i) A sincronização é um mecanismo pelo qual os agentes de uma determinada UP obtêm informações sobre valores de variáveis e diagnósticos identificados em outras UPs.

   ii) A negociação é um mecanismo que visa a garantir que ações não sejam efetivadas em uma UP sem garantir que elas não influenciarão negativamente no funcionamento de outra. Um exemplo disso ocorreria quando um poço aumenta a quantidade de fluido enviado para uma estação de coleta no mesmo instante em que esta enfrenta problemas para reduzir o nível dos seus tanques para poder aceitar tal aumento de demanda.

A solução usada para as interfaces de comunicação entre UPs está na plataforma JADE, adotada para a construção da PLA, uma vez que ela traz um serviço de entrega de mensagens entre agentes que residem em diferentes contêineres, como o caso de agentes de UPs diferentes. Sua descrição ficará para a seção sobre a implementação do SCAE.

Uma possível visão do campo após a implantação do SCAE é ilustrado na Figura 19.

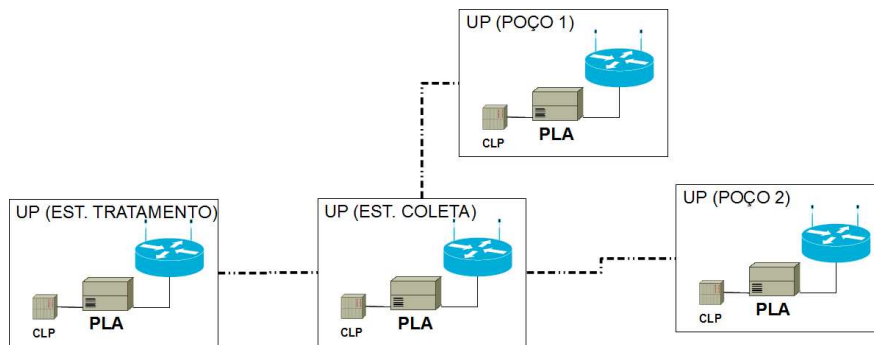



Figura 19: Visão do campo pós implantação do SCAE

4.2 PROJETO

As seções que se seguem descrevem os modos de operação da PLA, detalhamento da base de conhecimento, identificação dos agentes, detalhamento dos agentes e mecanismos de interação.

Sendo a PLA o componente chave do SCAE, buscou-se inicialmente estabelecer algumas premissas em relação ao seu projeto e construção:

a) A PLA deve ser implantada na própria UP;

b) Cada UP que estiver sob controle do SCAE deve receber uma instância da PLA;

c) A PLA deve ser implantada sem custos elevados e com o mínimo de alterações na estrutura de automação da UP;

d) Os agentes residentes em uma PLA necessitam da capacidade de interagir com os agentes de outra PLA;

e) Priorizar o funcionamento autônomo da PLA;

f) Buscar a redução na utilização da rede de comunicação entre as UP.

As premissas “a”, “b” e “c” limitam as opções de hardware disponíveis para a implantação da PLA. Em resposta a essa limitação, uma pesquisa foi conduzida com o objetivo de suprir o SCAE com uma plataforma de hardware adequada. Os resultados são apresentados em seção posterior. A premissa “d” reforça a idéia de cooperação e colaboração entre agentes em um SMA, e estabelece a noção de que o SMA representado pelo SCAE não significa apenas a interação entre os agentes de uma PLA, mas a interação global.

De acordo com a premissa “e”, quando duas funcionalidades ou recursos do SCAE forem conflitantes, ou trouxerem dificuldades tecnológicas para sua implementação deve-se optar por aquela que conduz a maior autonomia para a PLA.

4.2.1 Modos de funcionamento da PLA

A PLA possui dois modos de funcionamento: configuração e execução.

O modo de configuração é usado para que o operador do sistema possa alimentar a BC da PLA com os dados necessários para o seu funcionamento: especificações da UP, configurações e regras.

Nesse modo de funcionamento, todas as interfaces ficam indisponíveis, exceto as IHC de alimentação da base de regras, cadastramento das especificações e configurações do sistema.

No modo de execução, a PLA está instalada e em atividade na UP, ou seja, todas as suas interfaces estão abertas e os agentes, ativos.

4.2.2 Base de conhecimento

Conforme apresentado anteriormente, a PLA possui um conjunto de estruturas de dados, e cada uma armazena um conjunto distinto deles.

a) Especificações da UP

A base de dados de especificações foi construída para fornecer aos agentes uma visão sobre a UP onde reside e sobre as demais relacionadas a ela.

Esta base é formada por variáveis que descrevem o funcionamento da UP local e relacionadas, UPs relacionadas, diagnósticos que podem ser identificados na UP e ações de ajuste que podem ser aplicadas (Figura 20).

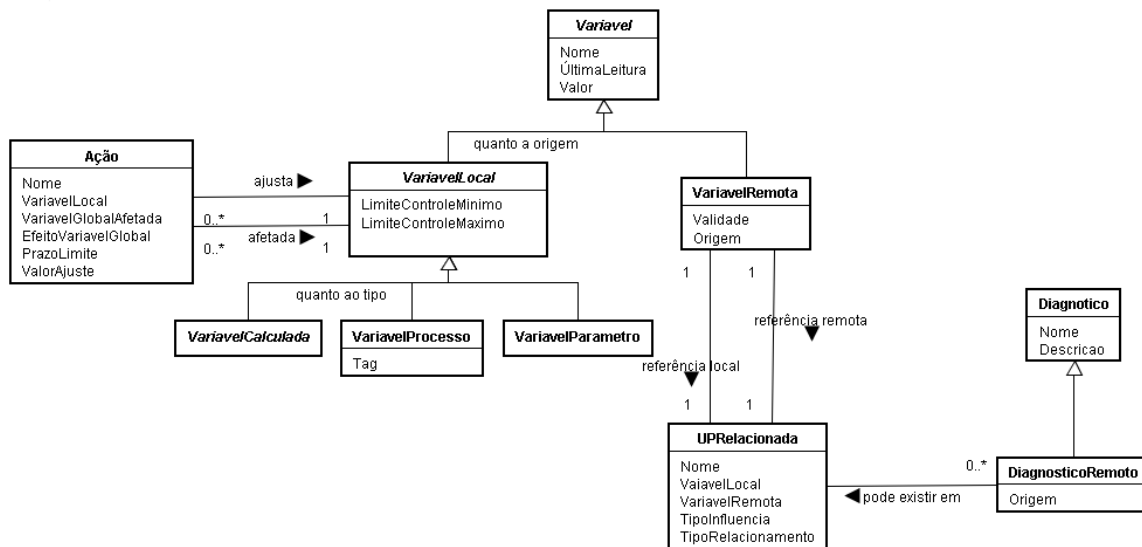



Figura 20: Estrutura da base de especificações da UP

A Tabela 2 apresenta as estruturas, seus atributos, tipo de dado do atributo e possíveis valores que poderá assumir (quando valores discretos).

Tabela 2: Estrutura da base de dados e especificações da UP

Nome da Estrutura	**Atributos**	**Significado**	**Tipo de dado**	**Possíveis valores**
Variável	Armazena as variáveis que representam a UP.			
	Tipo	O tipo da variável	Numérico	1-Variável de processo 2-Variável calculada 3-Parâmetro
	TipoOrigem	Se é parte da UP local ou de um UP relacionada	Numérico	1-Local 2-Remota
	Nome	Identificador da variável	Texto	
	Tag	Nome da *tag* correspondente no controlador	Texto	
	LimiteControleMinimo	Limite de controle mínimo para a variável	Número	
	LimiteControleMáximo	Limite de controle máximo para a variável	Número	
	UltimaLeitura	Data e hora da última leitura	Data e Hora	
	Valor	Último valor lido para a variável	Número	
	Histórico	(Ver bases de dados		

		operacionais)		
	Origem	UP de origem da variável, para o caso de variáveis sincronizadas de outras UP	UPRemota	Apenas quando a variável for remota
	Validade	O tempo de validade do valor da variável após ter sido adquirido.	Número	Apenas quando a variável for remota
UPRelacionada	UP que possuem algum relacionamento com a UP local			
	Nome	Identificador da UP	Texto	
	TipoInfluencia	O tipo de influência que exerce sobre a UP local.	Número	1-Influência 2-Depende
	VariavelLocal	Variável da UP local que a relaciona com a UP remota	Texto	
	VariavelRemota	Variável da UP remota que a relaciona com a UP local	Texto	
	TipoRelacionamento	O tipo de relacionamento das duas variáveis a local e a remota	Número	1-Direta 2-Inversa 3-Indefinido
Diagnóstico	Possíveis diagnósticos identificados na UP local ou em outras UP			
	Nome	Identificador do diagnóstico	Texto	
	Descrição	Descrição resumida do diagnóstico	Texto	
	Origem	UP de origem	Texto	Usado apenas para diagnósticos vindos de outras UP
Ação	Ação de ajuste que podem ser executadas na UP			
	Nome	Identificador da ação	Texto	
	VariavelLocal	Variável da UP a ser ajustada	Texto	
	VariavelGlobalAfetada	Variável que determina o relacionamento da UP local com outras, e poderá ser afetada pela ação. Ex.: Vazão bomba do poço X Nível do tanque da estação de coleta	Texto	
	EfeitoVariavelGlobal	O efeito que a ação causará na variável global (Item: VariavelGlobalAfetada)	Número	1-Aumento do valor 2-Diminuição do valor 3-Indefinido
	PrazoLimite	O prazo limite para execução da ação, ou seja, após sua	Número	

		identificação em quanto tempo ela deverá ser aplicada ao controlador.		
	ValorAjuste	Qual o novo valor para a variável.	Número	Este valor é fornecido quando a ação é selecionada para execução.

b) Memória de trabalho

A memória de trabalho foi projetada para permitir aos agentes obterem conhecimento sobre fatos presentes ou passados envolvendo variáveis da UP, diagnósticos e ações. Tais fatos são:

- Variáveis que tenham sido corretamente lidas desde a última consulta ao controlador;
- Variáveis de outras UPs que tenham sido corretamente sincronizadas e que não tenham expirado a validade;
- Diagnósticos identificados localmente;
- Diagnósticos recebidos de outras UP;
- Ações agendadas para execução;
- Ações executadas;
- Ações rejeitadas na negociação.

A estrutura completa da memória de trabalho é ilustrada na Figura 21.

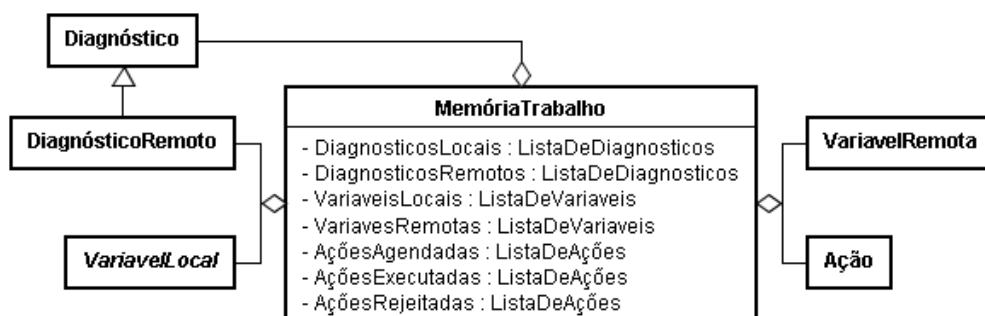



Figura 21: Estrutura da memória de trabalho

c) Regras

A base de regras é usada pelos agentes com o objetivo de inferir diagnósticos a partir da análise dos valores das variáveis da UP local ou relacionadas, ou diagnósticos identificados em outras UPs.

Sua construção é baseada na lógica de predicados, e foi definida para permitir a construção de regras que possam ser interpretadas pelo agente. As regras são formadas por <Resultado da Inferência> + <Expressões Lógicas>, onde:

i. Resultado da Inferência – é o fato que deverá ser gerado a partir da execução da regra, caso suas expressões lógicas tenham sido avaliadas como verdadeiras.
ii. Expressões Lógicas – quaisquer expressões válidas para a lógica de predicados.

O resultado da inferência será a identificação de um diagnóstico ou a seleção de uma ação de ajuste. Para que esse resultado possa ser interpretado pelo agente, dois predicados foram definidos:

a) diagnóstico(Nome): sendo Nome o identificador do diagnóstico na base de especificações da UP (Tabela 2).
b) ação(Nome): sendo Nome o identificador da ação na base de especificações da UP (Tabela 2).

Para compor expressões lógicas a partir de fatos existentes na memória de trabalho foram definidos os seguintes predicados:

i) variável(Nome, Valor, LimiteDeControleMáximo, LimiteDeControleMínimo): permite obter informações sobre as variáveis da UP.
ii) variável(Nome,Valor): é uma forma resumida do anterior.
iii) diagnóstico(Diagnostico): permite verificar a existência de diagnósticos anteriores na UP.
iv) ação_executada(NomeAção, LimiteDaFila): permite verificar a existência de ações que tenha sido identificadas anteriormente e que tenham sido efetivadas.
v) ação_negada(NomeAção, LimiteDaFila): permite verificar a existência de ações que tenham sido identificadas anteriormente e que tenham sido rejeitadas.

4.2.3 Definição dos agentes

Os agentes da PLA (Figura 22) são definidos a partir dos módulos que formam o MC do GCAD e são eles:

- Agente de Análise Local (AAL);
- Agente de Ajuste do Controlador (AAC);
- Agente de Verificação Continuada (AVC);
- Agente de Negociação (ANE);
- Agente de Sincronização (ASI).

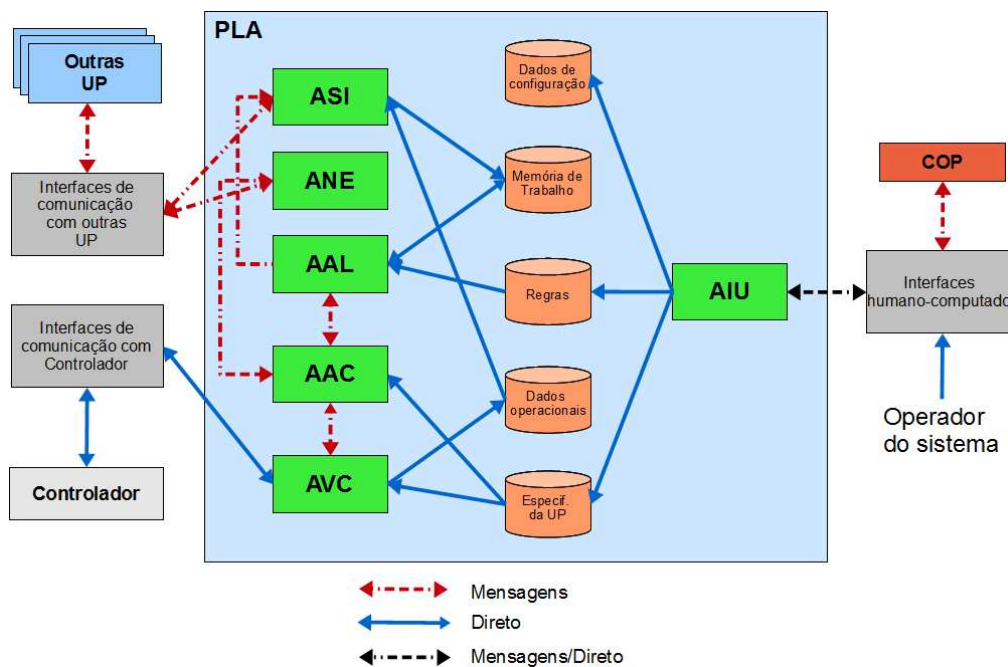



Figura 22: Distribuição dos agentes da PLA

As tarefas a serem realizadas por cada agente tomam como base as descrições do funcionamento dos módulos que estão descritos na seção 2.4.1 e detalhados em Pacheco (2011).

Na descrição do MC não são feitas especificações das IHC. Portanto, o Agente de Interface do Usuário (AIU) foi criado para esse fim e é responsável pelas ações de configuração, alimentação

da base de regras e cadastramento das especificações da UP. Ele representa os módulos complementares do GCAD, MGE e MTE.

Na Figura 22 é ilustrada a estrutura da PLA após a identificação dos agentes, assim como o relacionamento dos agentes com a BC, agentes locais e agentes de outras UPs.

4.2.4 Funcionamento dos agentes

Todos os agentes foram projetos utilizando o conceito de *Finite State Machine* (FSM). Essa abordagem facilita a construção de agentes com comportamentos fortemente dinâmicos, onde seu próximo estado vai depender de um estímulo resultante da execução de um estado anterior e não cabe a cada estado determinar qual o sucederá.

A descrição da FSM para cada agente foi feita utilizando um diagrama da UML[5] denominado Diagrama de Máquina de Estados.

a) Agente de Verificação Continuada (AVC)

Esse agente é responsável por ler os valores das variáveis da UP a partir do controlador e escrever novos valores para os parâmetros de controle deste, quando ações são identificadas pelo AAL.

Ao iniciar, (ver Figura 23) o agente consulta a base de conhecimento para obter as variáveis que deverá ler do controlador. A partir daí, periodicamente lê os valores das variáveis e os armazena na base de dados de processo e na memória de trabalho.

Paralelamente, ele aguarda uma mensagem com a solicitação de ajuste, quando for necessário alterar o valor de algum parâmetro no controlador.

Ao receber uma mensagem com a solicitação de mudança, ele tenta realizar o ajuste enviando o novo valor através da interface de comunicação e informa o resultado para o agente que solicitou o ajuste, se houve sucesso ou falha na escrita.

5 *Unified Modeling Language* (UML) é uma linguagem de modelagem utilizada para apoiar o desenvolvimento de sistemas baseados no paradigma da Orientação a Objetos.

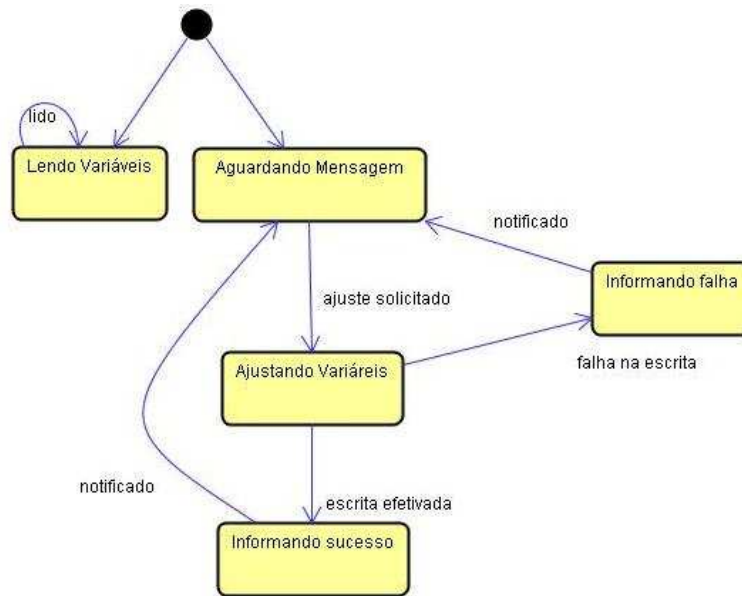



Figura 23: Máquina de estados para o AVC

b) Agente de Análise Local (AAL)

É responsável por verificar os limites operacionais das variáveis e, caso exista alguma fora do limite de controle, inicia o processo de análise tentando identificar um diagnóstico para as condições da UP.

Quando o agente identifica um diagnóstico, ele o insere na memória de trabalho e a análise continua, até que uma ação seja identificada ou toda a base de regras tenha sido avaliada (Figura 24).

Ao avaliar toda a base de regras sem encontrar nenhuma ação, conclui-se que os diagnósticos identificados são inconclusivos.

Então, o AAL solicita ao agente responsável pela sincronização do sistema a propagação do diagnóstico para células relacionadas. Ao encontrar uma ação, ele envia a mesma para o agente AAC responsável pela execução.

c) Agente de Sincronização (ASI)

Sincroniza os valores das variáveis das UPs relacionadas. O agente pode iniciar o processo de sincronização pela passagem do tempo, ou por solicitação de outro agente.

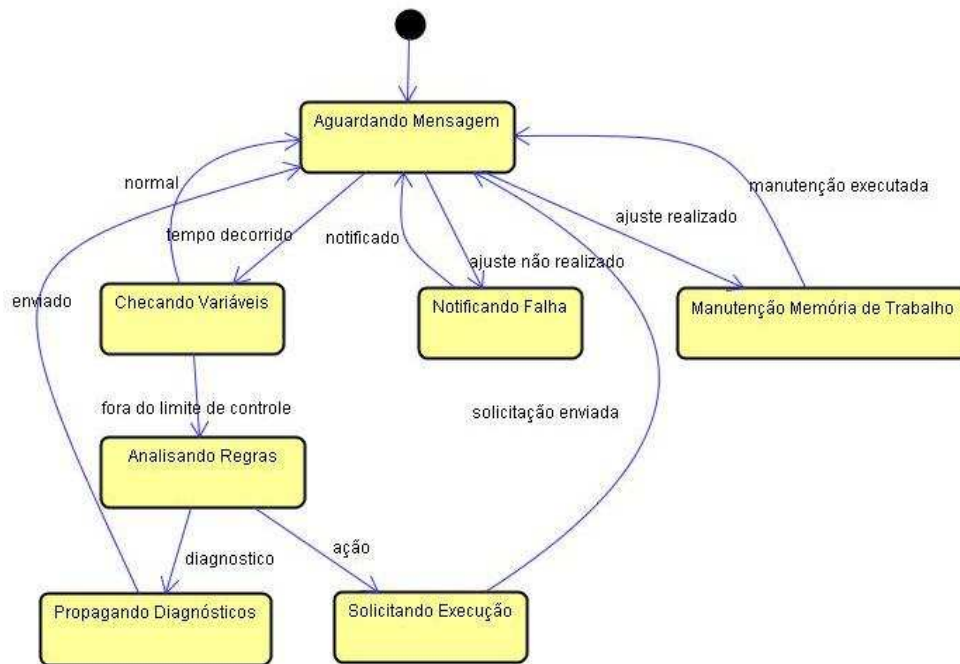



Figura 24: Máquina de estados para o AAL

Além de sincronizar as variáveis, o agente recebe os diagnósticos propagados de outras UPs para que sejam adicionadas a memória de trabalho local e, dessa forma, possam passar a ser utilizados na análise local.

Os detalhes de como ocorre a sincronização serão vistos na seção 4.2.6

d) Agente de Negociação (ANE)

O papel desse agente é negociar, com outros, alterações nas variáveis da UP que causem impacto nas demais. Seu funcionamento é descrito na Figura 25. Em seu ciclo de vida, ele aguarda até que uma solicitação de negociação seja encaminhada e, a partir daí, dá início ao processo de negociação enviando um pedido de ajuste para o ANE de cada UP relacionada.

Os detalhes de como ocorre a negociação serão vistos na seção 4.2.6 .

e) Agente de Atualização do Controlador (AAC)

O AAC é responsável por executar as ações identificadas pelo AAL. Ele verifica o impacto da ação em outras UPs. Na Figura 26 é ilustrado o funcionamento do agente.

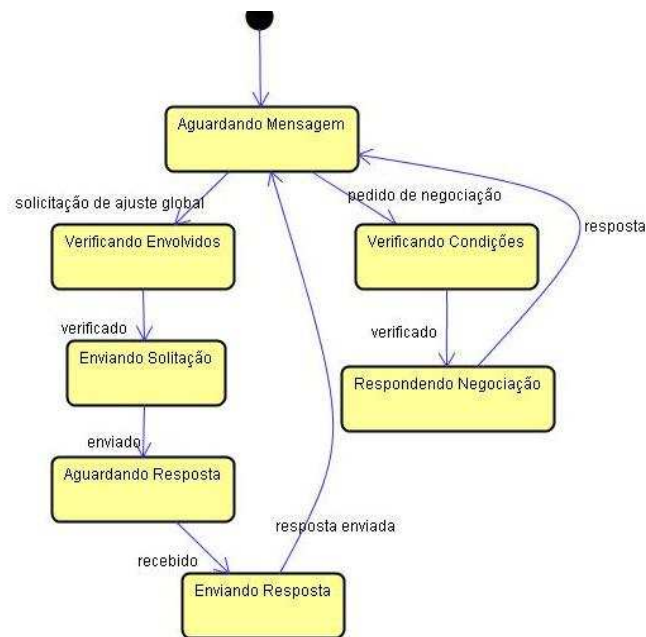



Figura 25: Máquina de estados para o ANE

Ao receber uma mensagem informando a identificação de uma ação de ajuste, o AAC verifica na BC o impacto da ação, e se esta necessita de autorização de outras UPs relacionadas. Caso seja necessário, ele solicita ao ANE para que proceda a negociação.

O agente deverá aguardar a resposta para repassá-la ao AVC. Se o ajuste não for autorizado,

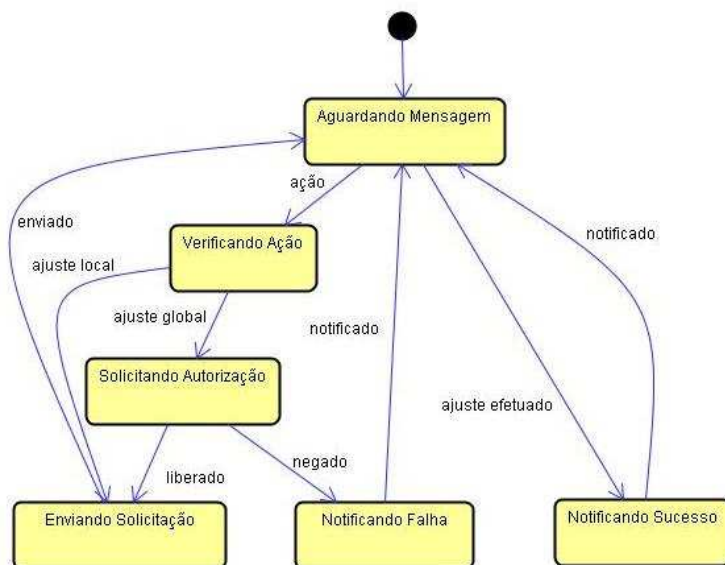



Figura 26: Máquina de estados para o AAC

o agente coloca a ação na lista de rejeitadas, em seguida notifica ao operador do campo, por meio do AIU, a existência de uma situação de anormalidade, cuja solução está além da sua capacidade. Se o tempo limite para execução da ação foi atingido, ela será mantida na lista de escalonadas para tentativa posterior e o operador será notificado por meio do AIU.

f) Agente de Interface do Usuário (AIU)

O AIU é responsável pela interação com os usuários, que podem ser: o operador do sistema, responsável por alimentar a BC e as configurações ou o operador do campo, responsável por acompanhar os diagnósticos e ações realizadas em cada UP. Ele é o único agente ativo quando a PLA está em modo de configuração.

Durante a execução da PLA, o AIU recebe as notificações dos demais agentes e as envia para o operador do campo. Dessa forma, o operador poderá acompanhar cada diagnóstico identificado, ação selecionada ou falhas na comunicação entre os agentes.

4.2.5 Interação entre os agentes da PLA

Os agentes residentes na PLA trabalham em constante cooperação, uma vez que a cada um foi atribuída uma tarefa em especial.

Sendo cada agente uma entidade de software autônoma, a distribuição de tarefas entre eles confere um maior paralelismo na execução das tarefas que formam a estrutura lógica de funcionamento da PLA e, consequentemente, são executadas em um menor tempo ante uma execução procedural e sequencial.

O diagrama da figura Figura 27 ilustra as interações que podem ocorrer entre os agentes de uma mesma PLA. Elas são representadas por setas horizontais, que representam as mensagens que um agente envia para outro, bem como a resposta esperada. As mensagens podem ser síncronas, quando o remetente precisa aguardar uma reposta para seguir suas tarefas, e assíncronas, quando o remetente não precisa permanecer bloqueado enquanto aguarda uma resposta.

A maior parte das interações é conduzida por meio de mensagens assíncronas, isso permite que o agente siga realizando outras tarefas enquanto aguarda a resposta de uma solicitação. Esse mecanismo também evita situações de *deadlock* no sistema, quando um agente fica aguardando por

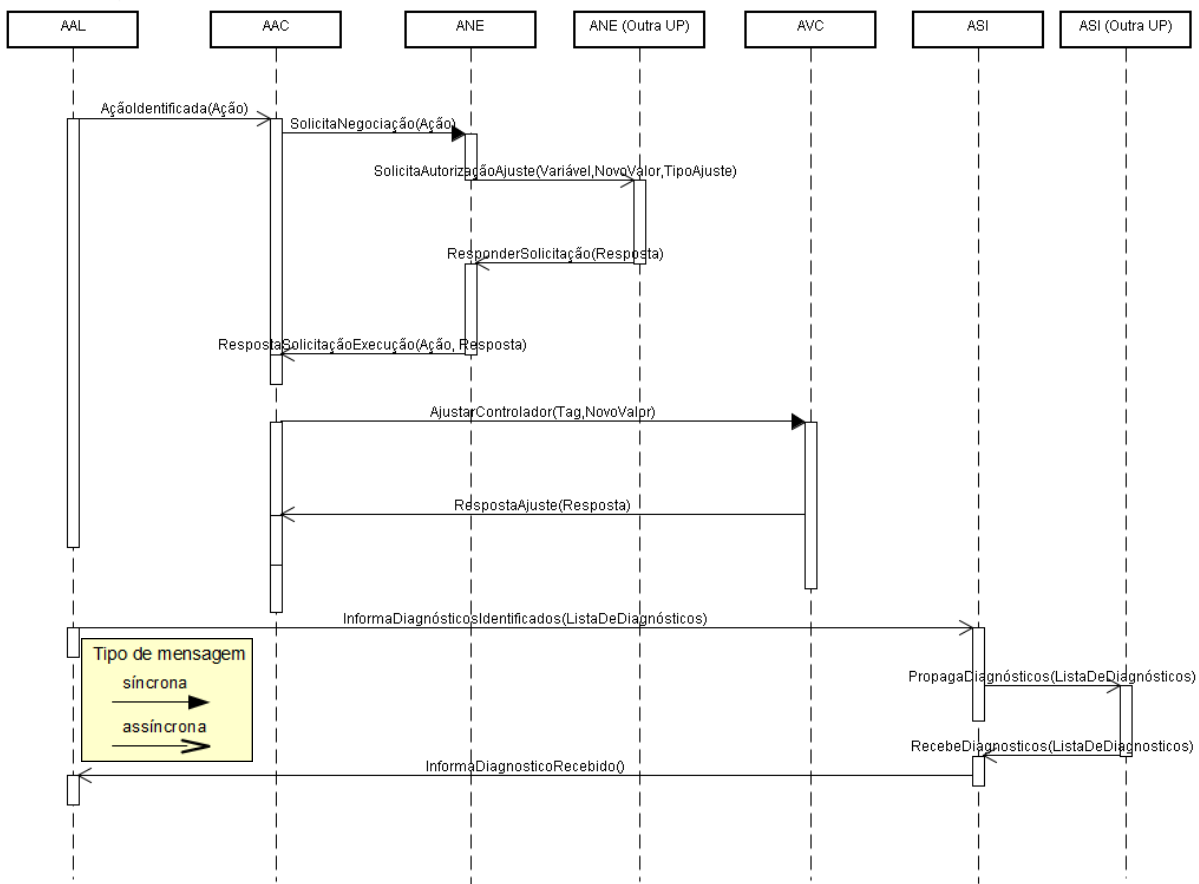



Figura 27: Interação entre os agentes da PLA

uma resposta de outro agente que foi bloqueado por ele.

A mensagem enviada pelo AAC ao solicitar a negociação para um ajuste com impacto em outras UPs é síncrona, pois as ações devem ser executadas na mesma ordem que são identificadas. Nesse caso, o agente não deve realizar outras atividades até concluir a primeira. O prazo limite de execução de uma tarefa impede que o agente aguarde indefinidamente por uma resposta, conforme descrito anteriormente.

4.2.6 Interação PLA x PLA

Como já foi dito anteriormente, as interações entre agentes em diferentes PLAs podem ser resumidas a dois mecanismos de cooperação - negociação e sincronização - e são melhor detalhados quanto ao seu funcionamento a seguir.

a) Negociação

A negociação é feita quando o AAC da UP recebe uma solicitação de execução de uma ação e este verifica que a sua execução pode influenciar o funcionamento de outra UP. Portanto, o ANE deverá ser notificado para que negocie o ajuste com as demais UPs envolvidas.

Para realizar a negociação, o agente:

1. Obtém a informação sobre qual variável será ajustada;
2. Verifica se o ajuste irá aumentar ou diminuir o valor da variável ou se não é possível definir o efeito que a ação trará sobre a variável;
3. Envia um pedido de ajuste para todas as UPs relacionadas que deverão:
    a. Observar os limites de controle de sua variável local que sofre impacto com a mudança que está sendo proposta.
    b. Responder ao agente solicitante.

Uma de três possíveis respostas são esperadas pelo agente:

- Se o ajuste foi autorizado, este é enviado ao controlador pelo AVC. Caso o ajuste seja corretamente enviado, a ação que resultou no ajuste é movida na memória de trabalho, passando da lista de agendadas para a de executadas.
- Se o ajuste for negado, a ação não será executada e será movida da lista de agendadas para a lista de ações rejeitadas da memória de trabalho.
- Se a resposta for "aguarde estabilização", a ação é mantida na memória de trabalho para execução futura (a ser comentada adiante).

As regras para responder à negociação são descritas a seguir, onde V_l é a variável que será ajustada, V_r é a variável remota que sofrerá impacto com a mudança em V_l *e* R é a relação entre as variáveis V_l e V_r . L_{min} e L_{max} são respectivamente os limites de controle mínimo é máximo da variável V_r. Dessa forma, o ajuste será liberado se, e somente se:

R1: se V_l for aumentada e R é direta e $V_r < L_{max}$

R2: se V_l for aumentada e R é inversa e $V_r > L_{min}$

R3: se V_l for diminuída e R é direta e $V_r > L_{min}$

R4: se V_l for diminuída e R é inversa e $V_r < L_{max}$

R5: se R é indefinida e $V_r > L_{min}$ e $< L_{max}$

Entretanto, antes de avaliar as regras acima, o agente verifica a existência de alguma ação escalonada que não tenha sido efetivada. Caso ocorra, a resposta será "aguarde estabilização", pois é necessário antes obter o efeito causado por aquela ação para só então inferir sobre as condições de operação da UP que está respondendo à negociação. A UP que iniciou a negociação irá manter a ação na lista, para uma tentativa posterior de negociação respeitando o tempo limite para efetivação da mesma.

O mecanismo de negociação foi assim especificado para impedir que as ações autônomas dos agentes da PLA levem o campo a situações de inconsistência, uma vez que, um ajuste local só poderá ser realizado quando houver garantia de que não ocorrerá efeito colateral nas demais UPs. Isso limita as ações por parte do SCAE, mas garante que não irá interferir negativamente no funcionamento do campo, pois, se o ajuste foi autorizado, é por que suas condições de funcionamento estão dentro dos parâmetros esperados e, portanto, ela poderá suportar um eventual aumento. Um exemplo disso é quando um poço deseja aumentar sua produção e a estação de coleta tem seu nível dentro dos limites para suportar o aumento da demanda de óleo.

b) Sincronização

A sincronização visa a alimentar os agentes com informações de outras UPs, valores de variáveis e estado de operação. É realizado pelo ASI da UP que deseja obter os valores das variáveis de outra UP e os ASIs das UPs relacionadas com a solicitante e que possuam informação sobre as variáveis desejadas.

Os passos para a sincronização são:

1. O agente envia uma mensagem para os ASI das outras UPs informando uma lista das variáveis que ele deseja;

2. Ao receber a mensagem, o agente verifica quais variáveis ele possui;
3. Caso possua alguma das variáveis solicitadas, ele:
    a. envia os valores e, junto com eles,
    b. envia uma lista de variáveis que ele também deseja;
4. O agente que iniciou a sincronização ao receber a resposta armazena os valores das variáveis recebidas;
5. Envia o valor das variáveis que foram solicitadas na mensagem de resposta do outro agente.

Note-se que este mecanismo foi assim criado para reduzir o número de mensagens para a sincronização das variáveis entre PLAs. Por esta razão, o agente que responde à sincronização aproveita a mensagem para também solicitar as que deseja. Isso evita que, instantes após ter respondido a uma sincronização, se inicie outra com o mesmo agente.

4.3 IMPLEMENTAÇÃO

Essa seção detalha a implementação do SCAE. Inicialmente, são apresentados alguns argumentos que justificam as soluções seguidas, com destaque para a linguagem usada para codificar a PLA e os componentes utilizados na implementação. Em seguida, é apresentada a documentação do software usando notações da UML (*Unified Modeling Language*) e as restrições que se impuseram na implementação.

4.3.1 Plataforma de hardware da PLA

As UPs típicas de campo, usadas pelo SCAE, utilizam um controlador que não suporta a execução da PLA, por conta das restrições de seu hardware e do software embarcado. Como esta é uma situação típica dos controladores industriais mais usados, uma solução de hardware precisava ser encontrada, a fim de viabilizar a adoção dos conceitos aqui propostos.

Por conta do pouco espaço disponível nos painéis onde o controlador é instalado, a pesquisa feita baseou-se em dispositivos com dimensões reduzidas, mas que resistissem às condições de

funcionamento a que seriam expostos no campo e, além disso, tivessem recursos de hardware (processamento, memória e comunicação) suficientes para a execução da PLA.

A PLA pode ser executada em qualquer dispositivo que suporte uma máquina virtual Java (ver seção seguinte) para dispositivos limitados, como *Smart Phones* e *Personal Digital Assistants* (PDA).

Os PDAs são construídos para substituir os computadores em tarefas que devem ser realizadas enquanto o usuário está em deslocamento. A maior parte deles suporta o uso em ambientes externos e, geralmente, são capazes de executar programas, tais como processadores de texto e planilhas de cálculo, além de suportar o armazenamento de grandes volumes de dados.

Para selecionar uma plataforma de hardware para a PLA, diversos PDAs foram analisados. O que se pôde observar é que, em geral, esses dispositivos apresentam recursos semelhantes, variando normalmente a robustez física. Dessa forma, dois modelos foram escolhidos para uma comparação com as características de um controlador usado na automação da maior parte dos poços nos campos de produção na Bahia. São eles o iPAQ hx2790 da Hewlett-Packard ™ e o XM65 da Janam. A principal diferença entre os dois é o grau de proteção, pois o XM65 é construído para suportar o uso intenso em campo, enquanto o iPAQ é destinado para uso doméstico, como pode ser visto na Tabela 3.

Tabela 3: Comparação Controlador x PDA

	Controlador	**iPAQ Pocket PC hx2790b/HP**	**XM65/Janam**
Processador/Clock	14.5476 MHz x 2	624 MHz	226 MHz
Memória	2 x 256 Kbytes	384 MB	64MB SDRAM 128 MB NAND
Temperatura de operação	0.. 60 Cº	0.. 40 Cº	-20.. 50 Cº
Umidade relativa	<= 80% sem condensação	<= 90%	<= 90% sem condensação
Grau de proteção	IP30	não informado	IP54 Categoria II
Interfaces de comunicação	Serial RS232 e RS485	Serial RS232 USB 2.0 Wi-fi (802.11b)	Serial RS232 USB 2.0 Wi-fi (802.11b/g)
Sistema Operacional	-	Windows Mobile ™ 5.0	Microsoft® Windows Mobile® 5.0 Premium

Os recursos computacionais (memória, processamento e armazenamento) dos dois PDAs são muito superiores aos do controlador. A Memória de armazenamento é um item importante, pois na PLA ela deve permitir o armazenamento de dados históricos sobre o processo.

A temperatura de operação, bem como o grau de umidade suportado, é semelhante nos três aparelhos.

Já o grau de proteção, que determina a quais condições o aparelho pode ser exposto, é ainda superior no PDA da Janam, em relação ao controlador. O fabricante do iPAQ não especifica o grau de proteção do mesmo.

Para validação do conceito do SCAE, uma instância da PLA foi então implantada em um PDA iPAQ, que foi selecionado por apresentar os recursos necessários e estar disponível no mercado nacional, o que facilitaria sua aquisição. Além disso, por se tratar de uma pesquisa de caráter acadêmico, ela não previa a exposição do sistema ao ambiente real dos campos de produção, portanto, qualquer investimento nesse sentido não traria benefícios à pesquisa. Detalhes desta implantação e a maneira como ele é utilizado serão descritas no próximo capítulo.

4.3.2 Plataforma de software da PLA

Um importante passo na implementação de um sistema é a escolha da linguagem com a qual ele será codificado. Ela determinará, entre outras coisas, os desafios encontrados na codificação e a eficiência do sistema construído.

Uma maneira de selecionar uma linguagem de programação é observar a natureza dos elementos relacionados ao sistema e ao ambiente onde o mesmo está inserido, incluindo usuários, hardware, obstáculos legais ou comerciais e desafios tecnológicos.

Para a implementação do SCAE foram considerados dois aspectos importantes da estrutura de um campo de produção de petróleo (como discutido no capítulo 2):

i. A heterogeneidade dos equipamentos utilizados na automação do campo(controladores, computadores, redes de comunicação);

ii. A diversidade de padrões necessários para garantir a interoperabilidade entre os diferentes equipamentos, sistemas ou redes de comunicação.

Assim, unindo esses aspectos às premissas de projeto (tópico 4.2), e considerando os direcionamentos sobre o uso de linguagens e padrões abertos para a construção de sistemas[6], a linguagem utilizada deverá cumprir os requisitos a seguir:

1. Ser de código fonte aberto;
2. Ser construída sob especificações de padrões aberto;
3. Ser portável.

Entre as linguagens mais difundidas hoje, Java e C++ se encaixam nesses requisitos. Java foi escolhida por sua simplicidade de codificação, assim como a maior parte das linguagens orientadas a objeto e a existência de ambientes de desenvolvimento maduros e de distribuição livre, além da grande oferta de componentes que permitem poupar esforços na codificação. Adicionalmente a isso, um programa escrito em Java pode ser executado tanto em um computador de mesa como em um dispositivo, tal qual especificado para a PLA. Basta, para isso, ser construído seguindo as especificações J2ME - Java 2 *Micro Edition* (SUN, 2010). O mesmo não acontece com a linguagem C, por exemplo, onde o programa executável deve ser gerado utilizando um compilador e as bibliotecas específicas para o dispositivo onde será executado.

A linguagem .NET é também uma linguagem aberta e portável. Entretanto, os ambientes de desenvolvimento disponíveis são proprietários, o que desencoraja o seu uso acadêmico.

Após a escolha da linguagem, o passo seguinte foi a escolha de componentes que pudessem encurtar o caminho da codificação do SCAE.

4.3.3 Componentes utilizados

A escolha dos componentes deve considerar, primeiramente, compatibilidade com a Java e os aspectos que levaram a sua escolha, visando a assegurar a manutenção e a evolução do sistema.

O primeiro deles refere-se à construção dos agentes. Conforme visto no capítulo 3, construir SMA significa projetar entidades de software que sejam autônomas, tenham capacidade de raciocínio baseado em conhecimento e possam comunicar-se através da troca de mensagens em nível de conhecimento. As linguagens orientadas a objeto, como é o caso da Java, não oferecem em sua definição original meios para lidar com esses aspectos. Assim, a plataforma JADE foi

6 Este trabalho procurou ser aderente aos requisitos de projeto de software preconizados pela Petrobras.

selecionada para a construção dos agentes, pois, além de ser escrita em Java, ela é aberta, portável e utiliza padrões abertos para a construção de seus serviços. A plataforma JADE é apresentada em mais detalhes no Apêndice A.

Apesar de representar um importante ambiente para a construção e execução de SMA, JADE concentra-se nos serviços de ciclo de vida dos agentes e na comunicação entre eles, não oferecendo suporte nativo para outras necessidades do SCAE, tais como representação do conhecimento, armazenamento persistente de dados e comunicação com o controlador, o que foi preenchido com o auxílio de dois outros componentes: TuProlog (também melhor especificados no Apêndice B) e DB4o.

A biblioteca TuProlog foi utilizada para implementar o mecanismo de raciocínio do AAL, e sua escolha foi motivada por ser compatível com J2ME. Vale destacar que não foram encontradas alternativas de mecanismos de inferência compatíveis com a especificação J2ME.

O DB4o é um banco de dados orientado a objetos e permite salvar objetos Java em arquivos. Ele foi utilizado para armazenar os dados da base de conhecimento dos agentes e também é compatível com J2ME.

Na Tabela 4 pode ser visto um resumo das características principais dos componentes selecionados. Na Figura 28, é ilustrada cada tecnologia aplicada na implementação da PLA, e como elas se sobrepõem.

Tabela 4: Síntese dos componentes selecionados

Nome	**Versão**	**Finalidade**	**Distribuição**
JADE	3,7	Biblioteca Java para codificação dos agentes e ambiente de execução.	Free/Open Source http://jade.tilab.com
TuProlog	2	Biblioteca Java que implemente um motor de inferência para a linguagem Prolog.	Free/Open Source http://alice.unibo.it/xwiki/bin/view/Tuprolog
DB4o	7,4	Biblioteca Java para o armazenamento de objetos através em arquivos.	Free/Open Source http://www.db4o.com

Java recobre praticamente toda a implementação da PLA, a exceção de parte da interface de comunicação com o controlador que dependerá do *driver* de comunicação utilizado por cada fabricante. Isso se deve ao fato dos controladores utilizarem protocolos de comunicação próprios.

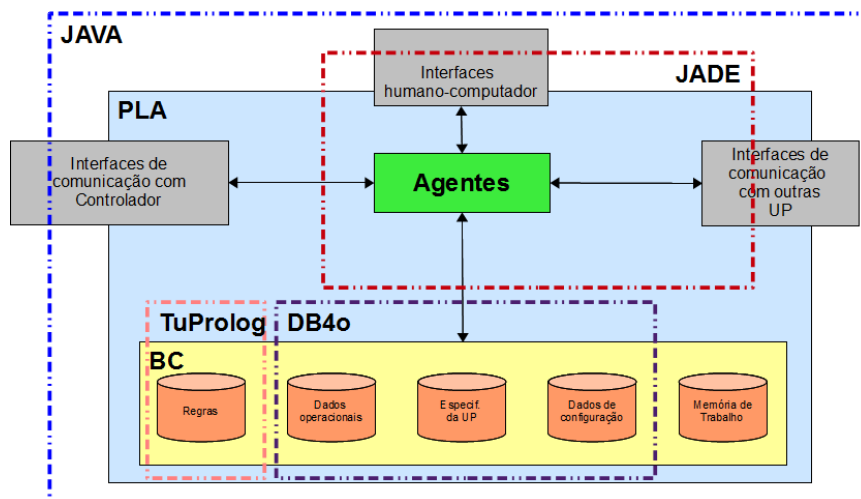



Figura 28: Sobreposição das tecnologias na PLA

Portanto, para se comunicar com diversos controladores, o SCAE especifica uma interface que deverá ser implementada para acesso ao *driver* específico do controlador.

Em seguida, a plataforma JADE fornece as classes que constroem os agentes e os serviços de comunicação entre agentes de diferentes contêineres. Quando JADE é utilizado em um ambiente J2ME, um protocolo denominado *Jade Inter Container Protocol* (JICP) é utilizado. Ele foi escrito exclusivamente para lidar com os requisitos das redes sem fio.

4.3.4 Organização dos arquivos

Todos os arquivos que formam o software da PLA são organizados em três subdiretórios, dentro do diretório raiz da aplicação, e são eles:

- *config*: contém os arquivos de configuração da PLA;
- *db*: contém o arquivo usado pelo DB4o para o armazenamento persistente das bases de dados;
- *rules*: contém os arquivos com extensão .pl que formam a base de regras da PLA;
- *lib*: contém os arquivos binários das bibliotecas utilizadas.

4.3.5 Codificação do software

Os principais pontos da codificação do SCAE são apresentados sob a forma de Diagrama de Classes da UML, que representa a estrutura das classes implementadas. Alguns diagramas foram simplificados para permitir sua visualização.

A documentação está dividida em: interfaces de comunicação, estruturas da BC e agentes.

1) Interfaces de comunicação

A interface de comunicação com o controlador tem o papel de isolar, do agente, a forma como ele lê ou escreve dados no controlador. Entretanto, cada controlador pode usar os seus próprios protocolos de acesso. Por essa razão, sua implementação final dependerá do controlador utilizado. Ela foi codificada como uma interface Java, que é um tipo especial de classe que contém apenas a definição dos atributos e métodos a serem implementados.

A estrutura da interface *IDataAccess* (Figura 29) é composta de quatro métodos:

- *connect()* – método usado para iniciar a ligação com o controlador;
- *disconnect()* – encerra a ligação com o controlador, se *keepConnect* é falso sua execução ocorre ao final de cada leitura;
- *read(variables: Map)* – recebe uma lista de variáveis a serem lidas do controlador;
- *write(variable: ProcessVariable)* – recebe a variável com o novo valor para ser escrito no controlador.

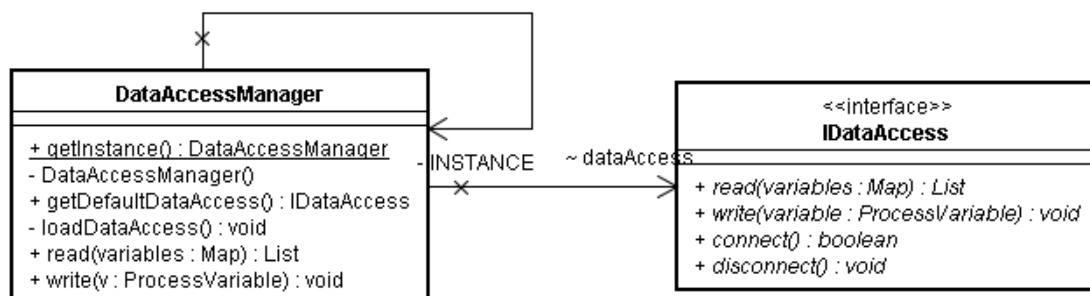



Figura 29: Estrutura da interface *IDataAccess*

A classe *DataAccessManager* é responsável por criar uma instância da classe real que implementa a *IDataAccess* com o propósito de integrar o *driver* de acesso ao controlador ao software da PLA.

Para o armazenamento das configurações da PLA foram utilizados os recursos da classe *java.util.Properties,* que fornece os métodos necessários para o armazenamento de parâmetros de configuração da aplicação por meio de arquivos texto.

Os parâmetros são armazenados e recuperados por meio de um par chave+valor, onde chave é um nome identificador para o parâmetro e valor o conteúdo que será atribuído a ele.

A adição ou alteração de parâmetros pode ser feita editando um arquivo denominado *system.properties* que fica localizado na pasta *config* do diretório de instalação do software.

2) Estruturas da BC

Para encapsular os métodos de manipulação das bases, foi criada a classe *DBManager* (Figura 30). Ela permite isolar do código do agente a forma de armazenamento dos dados. Essa estratégia foi utilizada para garantir que a forma de armazenamento possa ser substituída sem causar impacto em outras classes do sistema.

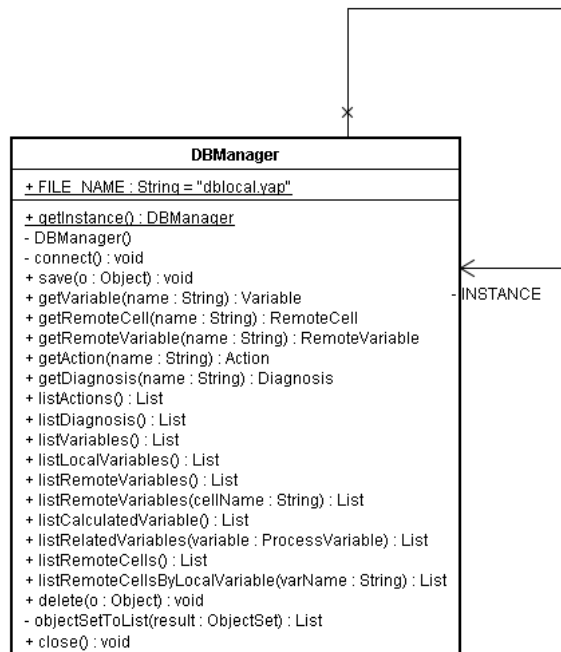



Figura 30: Estrutura da classe *DBManager*

A base de dados de especificações da UP e a base de dados operacionais foram codificadas como objetos Java, e são persistidos na memória de armazenamento da PLA através da biblioteca DB4o.

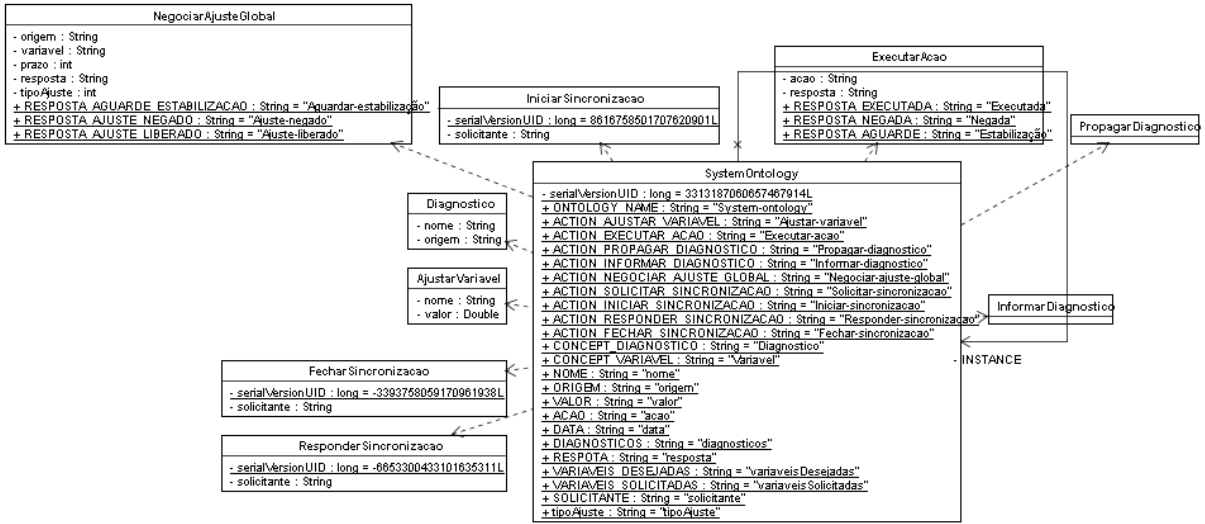



Figura 31: Ontologia do SCAE

A memória de trabalho é implementada pela classe *WorkMemory* e sua estrutura é ilustrada na Figura 32.

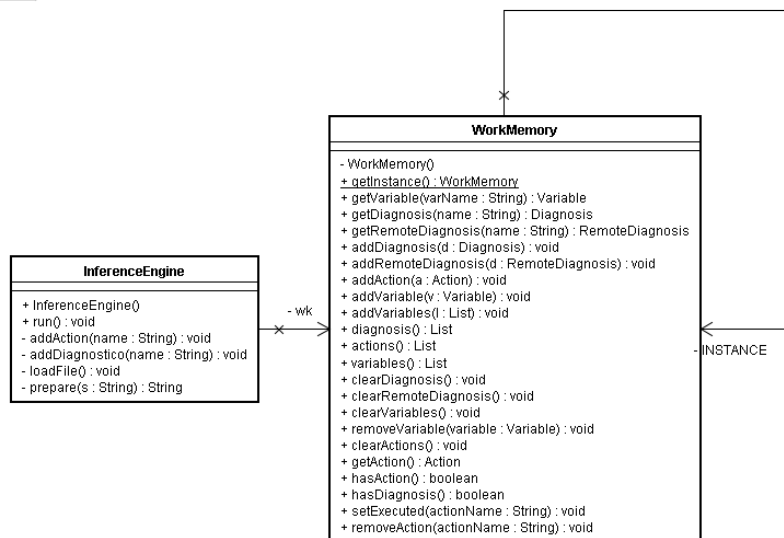



Figura 32: Estrutura da *WorkMemory*

A base de regras é implementada por meio da biblioteca TuProlog e as regras são fornecidas por meio de um arquivo texto com extensão .pl. Cada arquivo contendo as regras é lido e interpretado pela classe *InferenceEngine,* e é através dela que o AAL infere os diagnósticos ou ações. Seus principais métodos são:

- *loadFile()*: carrega os arquivos contendo as regras para a base de regras do sistema;
- *run()*: Inicia o processo de inferência avaliando as regras. Quando um diagnóstico ou ação são encontrados, são enviados para a memória de trabalho.

3) Agentes

Cada agente é uma classe que estende a classe *jade.core.Agent*, assim como seus comportamentos que devem estender as classes, *OneShotBehaviour* e *FSMBehaviour.* A documentação completa dos agentes é encontrada no Apêndice F.

O diagrama da Figura 31 ilustra a estrutura completa da *SystemOntology* classe que representa o vocabulário do SCAE.

A comunicação entre os agentes é feita através da troca de mensagens por um recurso nativo da plataforma JADE. Entretanto, para que uma mensagem possa ser decodificada pelo seu destinatário, um vocabulário comum deve ser usado. Sua implementação precisa estender a classe *Ontology* do JADE.

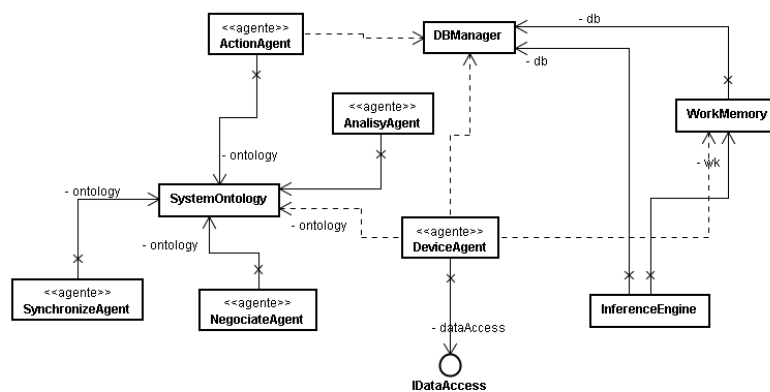



Figura 33: Agentes e demais classes

As classes que implementam os agentes, bem como a relação com as demais, descritas acima, são ilustradas na Figura 33. Agentes estão marcados com o esteriótipo[7] <<*agente*>>.

Alguns pontos da codificação foram omitidos, pois se referem a operações que não estão diretamente ligadas ao funcionamento do SCAE e não agregariam valor à documentação.

O AIU não foi implementado na versão atual do SCAE, pois, a validação focou no conceito de autonomia e resolução distribuída de problemas do mesmo. A participação do operador não precisaria, portanto, ser avaliada.

7 Em UML um esteriótipo é uma marcação delimitada pelos caracteres <<>> e, usada para acrescentar novas características aos elementos dos diagramas.

5 VALIDAÇÃO DA PROPOSTA

Este capítulo apresenta os experimentos realizados com o objetivo de validar a proposta do SCAE.

Espera-se, com esses experimentos, demonstrar como a utilização da abordagem proposta pelo SCAE pode melhorar a eficiência das tecnologias existentes para o gerenciamento de campos de produção de petróleo, uma vez que sua estrutura distribuída permite transformar cada unidade da produção (UP) em uma unidade autônoma e com capacidade de comunicação com as demais. Dessa forma, o diagnóstico sobre as condições de cada UP pode ser feito localmente sem a necessidade de envio dos dados para a COP, o que reduz o tempo entre a ocorrência do problema, sua identificação, e a aplicação de ações para ajustar o funcionamento. A troca de informações de diagnósticos entre as UPs também permite a identificação e tratamento de situações que antes dependiam inteiramente da participação do operador.

Essa abordagem transforma o campo de produção em um sistema distribuído (SD) composto de várias unidades que se interrelacionam. A partir desse relacionamento emerge um comportamento inteligente global, uma vez que o ajuste do funcionamento de cada UP é primeiramente baseado em uma avaliação de suas condições locais, acrescido de análise das condições locais das demais UPs que, de alguma forma, possam influenciar no seu funcionamento local.

5.1 ESTRUTURA LABORATORIAL

Para validação da proposta, foram utilizadas as instalações do Laboratório de Elevação Artificial (LEA) da Universidade Federal de Bahia (UFBA), que conta com um Sistema de

Bombeio Mecânico (SBM) completo, bem como as demais instalações e utilidades necessárias à operação de um campo de produção. O SBM instalado no LEA é completamente instrumentado e com o fundo do poço acessível e visível para servir de suporte laboratorial, visando a validar modelos existentes e embasar experimentalmente estudos de fluido dinâmica.

Todos os componentes do SBM são industriais e em escala real, com exceção da profundidade (menor no poço do LEA). Dessa forma, este SBM reproduz a maioria das condições operacionais encontradas nos campos de petróleo (BARRETO FILHO *et al.*, 2008).

Essa planta conta também com instrumentação para medição e controle de todas as variáveis importantes para o processo. O controle é exercido por um CLP (Compact Logix, da Rockwell Automation), capaz de executar as operações e lógicas exigidas, além de um sistema para supervisão, controle, coleta e registro dos dados (Figura 34).

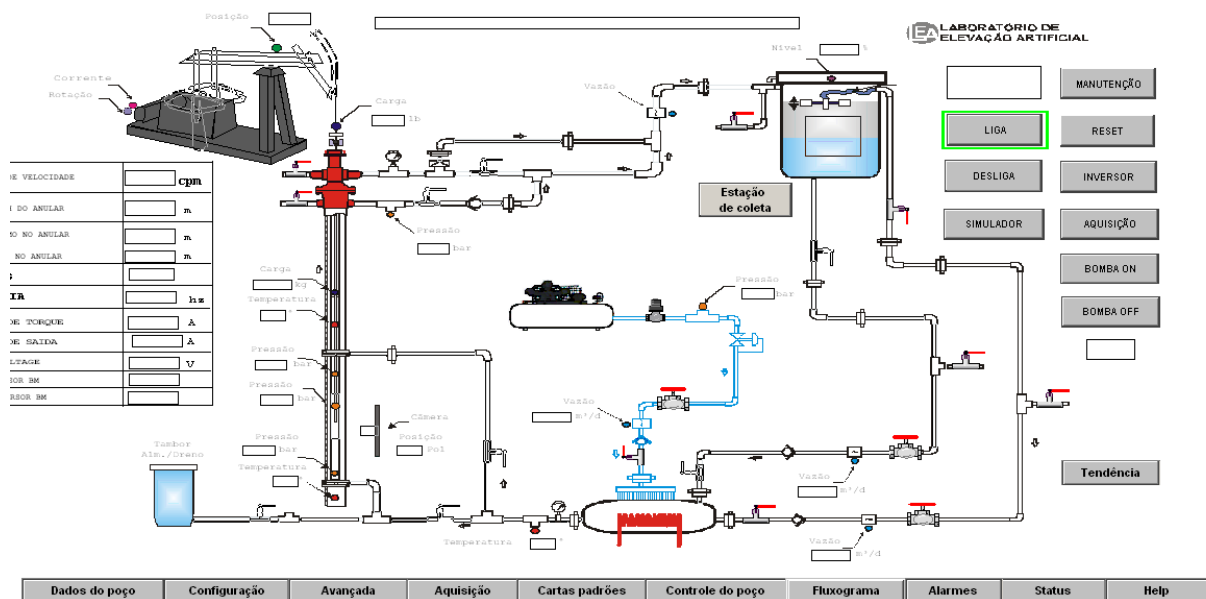



Figura 34: Tela do supervisório do LEA

Por questões de metodologia de trabalho, durante a idealização do laboratório e no projeto conceitual, a área de campo foi separada em algumas partes (BARRETO FILHO *et al.*, 2008):

i. Poço: poço de produção de petróleo propriamente dito, onde se encontra o SBM e seus componentes;

ii. Tanque de Armazenagem: seção pós-produção, para onde é encaminhado o fluido produzido. Esse tanque funciona também como unidade separadora;

iii. Retorno: mecanismo que torna o processo fechado (reciclando os fluidos para reutilização);

iv. Reservatório: conjunto de equipamentos responsáveis pela simulação das características do reservatório, no que se relaciona à interface com os poços.

5.2 DESCRIÇÃO DO AMBIENTE

Um ambiente foi construído visando à representação em laboratório das etapas de elevação, coleta e tratamento em um campo de produção (Figura 35). Essas etapas são representadas, respectivamente, por: poço equipado com BM (PCO), estação de coleta (ECO) e estação de tratamento (ETO). Vale ressaltar que o funcionamento do SCAE é independente do método de elevação utilizado no poço. O BM foi adotado para o experimento, por já estar em operação no LEA e contar com toda a instrumentação para operá-lo, além de ser o mais representativo pela quantidade e qualidade de situações que podem ser simuladas.

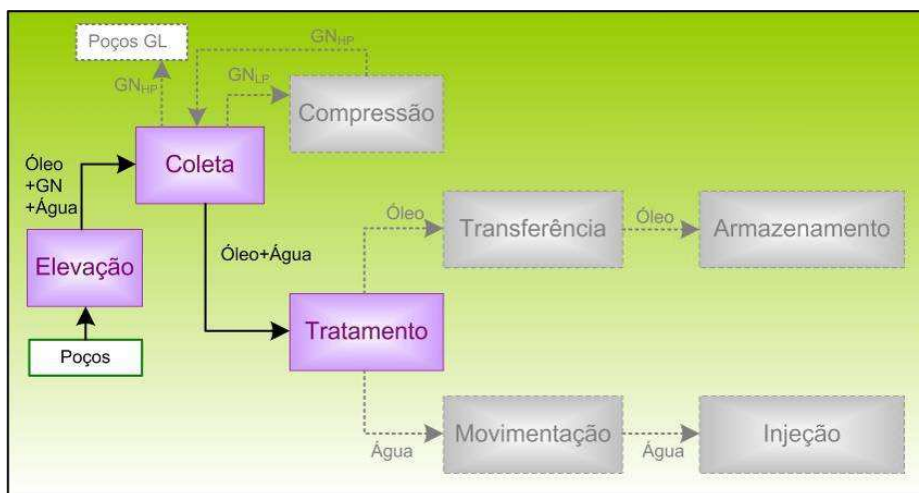



Figura 35: Etapas da produção analisadas

Para a execução do sistema, cada uma das UPs recebeu uma instância da PLA. O PCO foi representado pelo SBM do LEA, e as variáveis utilizadas no experimento são: nível dinâmico do poço, obtido através de transmissor de pressão; ciclos por minuto (CPM) da UB, obtido através do inversor de frequência; e pressão de fluxo, obtida através de um manômetro instalado do fundo do poço.

A ECO foi implementada utilizando o tanque de armazenamento do SBM do LEA e é equipado com um sensor de nível ultrassônico.

Linhas de retorno ligam o tanque de armazenamento ao fundo do poço do SBM do LEA para que o óleo possa circular de maneira cíclica no sistema. Esse nível, assim, é manipulado de acordo com os planos de experimento para as variáveis de fundo do poço e das ECO e ETO. Em um poço no campo, esse nível seria ditado pela vazão do reservatório que o alimenta, e pela capacidade de armazenamento e processamento das estações de tratamento e de coleta.

Para que o experimento pudesse refletir mais fielmente as condições reais de campo, foi criado um controle do nível virtual do tanque da ECO. Dessa forma, quando a recirculação precisa ser ligada para que o óleo alimente o fundo do poço, o nível do tanque da ECO não é afetado, o que ocorre apenas com o acionamento das bombas que conduzem o fluido até a ETO.

Um programa em Ladder, embarcado no próprio CLP, é responsável por ler o sensor de nível do tanque, implementar o nível virtual da ECO, bem como acionar as três bombas utilizadas para movimentação do fluido até a ETO.

A Figura 36 ilustra a dinâmica do processo envolvendo as UPs, PCO, ECO e ETO. Vê-se que, mesmo quando o nível do anular do PCO aumenta, o nível do tanque de armazenamento da ETO se mantém estável, só alterando quando as bombas são acionadas.

Como o sistema conta com apenas um tanque de armazenamento, a ETO foi completamente implementada em Ladder e embarcada em outro CLP (Compact Logix, da Rockewll Automation) e se baseia, por sua vez, na variação do nível da ECO, para determinar a vazão de entrada na ETO (Figura 36).

O óleo produzido pela UB é direcionado para o tanque de armazenamento, e conforme descrito anteriormente, é redirecionado para o reservatório.

As variáveis utilizadas na ECO e ETO são, portanto, o nível do tanque e situação de operação de cada bomba (*on*/*off*).

As PLAs das UPs, ECO e ETO foram instaladas, cada uma numa instância virtualizada de um computador, com a seguinte configuração: 128 MB RAM, 10 GB HD e sistema operacional Windows™. Para a comunicação entre cada PLA e o CLP, foi utilizado, como *driver*, o padrão OPC [8] implementado pela biblioteca JeasyOPC[9] (exceto para a PCO).

A infra-estrutura de hardware e software (Figura 37) que deu suporte à simulação foi

8 OPC (*OLE Process for Control)* padrão de comunicação para aplicações industriais, baseado no padrão DCOM, alternativa da Microsoft para a comunicação entre objetos distribuídos CORBA.

9 JEasyOPC é uma biblioteca escrita em linguagem Java que implementa o padrão OPC, seu código fonte pode ser encontrado em: http://www.opcconnect.org.

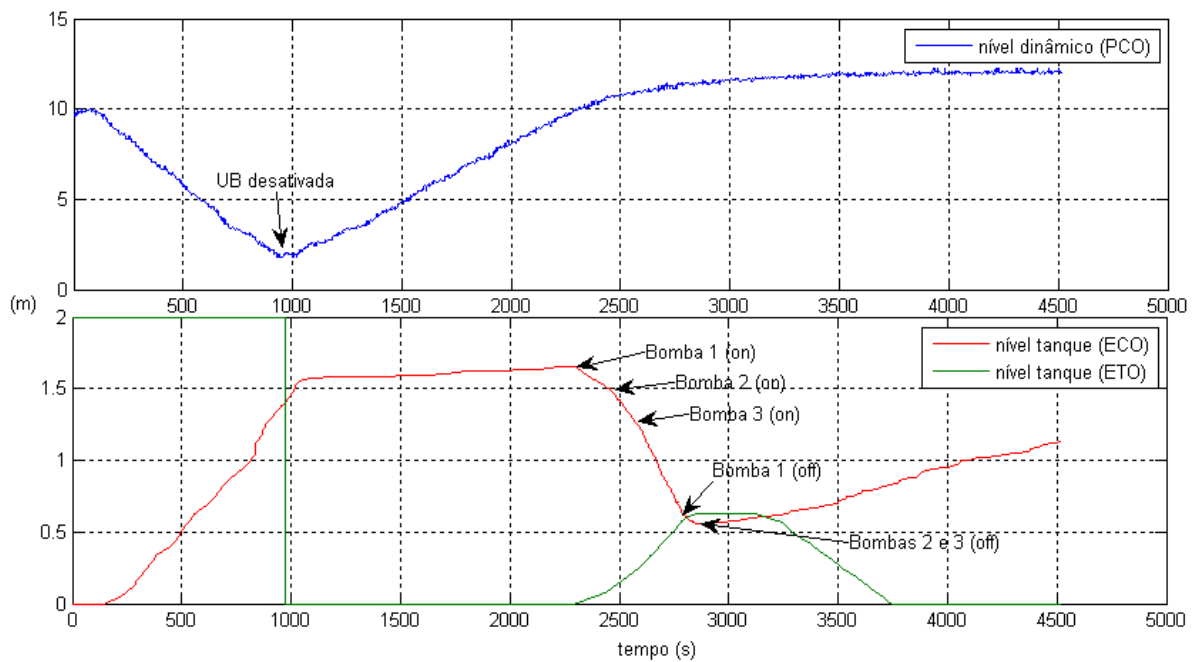



Figura 36: Comportamento das variáveis de UP PCO, ECO e ETO

composta por:

i. Computador com as seguintes configurações:

- Sistema Operacional: Windows Vista Bussiness ©
- Processador: Intel Core 2 Duo 1,8 GHz
- Memória RAM: 2 GB
- Espaço para armazenamento em disco: 160 GB
- Conexão *wireless*

ii. SBM – LEA

iii. 2 CLPs Compact Logix da Rockwell Automation

iv. 1 CLP SLC 500 da Rockwell Automation

v. 1 Roteador wireless 802.11g

vi. Software JavaSniffer (Rockwell Automation) para captura das mensagens trocadas entre os agentes

vii. Software Sun™ Virtual Box da Sun Microsystems para construção das

máquinas virtuais

viii. Software Logix 5000 (Rockwell Automation) para construção do programa de controle da ECO e ETO

As máquinas virtuais foram utilizadas para simular computadores, nos quais a PLA para as UPs, ECO e ETO foi implantada. O CLP da PCO foi interligado ao CLP da ETO por meio de uma rede *DeviceNet.* Os demais foram ligados diretamente ao roteador e utilizam, para isso, o protocolo Ethernet/IP.

A instância da PLA que controla a PCO foi implantada no PDA especificado na seção de

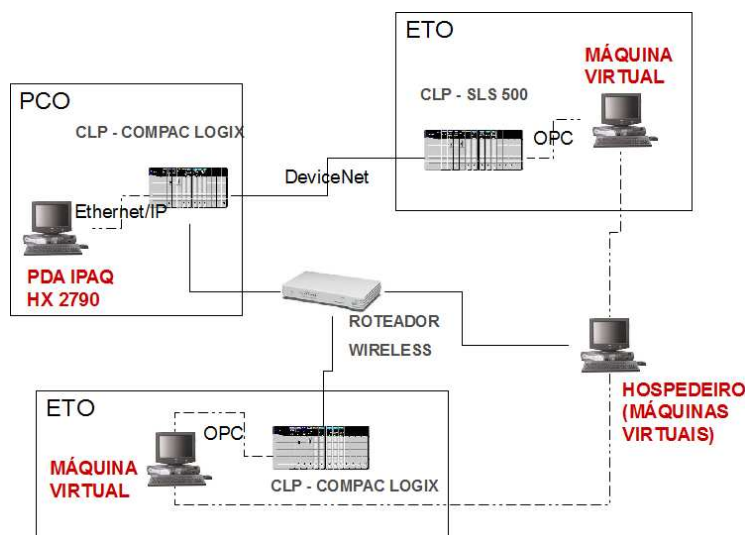



Figura 37: Infra-estrutura de hardware

projeto do capítulo anterior. Isso permitiu validar o funcionamento do sistema no ambiente de hardware que o mesmo deverá encontrar em uma situação real.

Como o sistema operacional do PDA não suporta a execução do padrão OPC, uma alternativa foi a utilização de uma biblioteca que realiza o acesso aos dados do controlador. A biblioteca NetLogix permite o acesso a *tags* em controladores da linha Compact Logix. Por ser codificada em .Net, foi necessário ainda escrever um *driver* para acesso por meio do SCAE, já que o mesmo foi desenvolvido usando a linguagem Java.

5.3 CASOS DE TESTES E RESULTADOS OBTIDOS

Os experimentos descritos a seguir foram conduzidos objetivando demonstrar três principais funcionalidades do sistema, que são:

a) A capacidade de análise local das condições da UP e aplicação de ações para a correção de situações de anormalidade;

b) A capacidade de realizar o diagnóstico distribuído das condições operacionais do campo (teste do mecanismo de sincronização);

c) A capacidade de negociação entre as UPs visando a solução de problemas locais, mas que exercem influência global no sistema.

Para a demonstração dessas habilidades, o comportamento das UPs, através de suas principais variáveis, foi capturado e organizado em gráficos sobrepostos, tendo o tempo como fonte de dados do eixo X.

A interação entre os diversos agentes do sistema foi capturada por meio da ferramenta JavaSniffer.

Para a condução dos experimentos, a BC de cada PLA foi alimentada com informações sobre o processo no qual deve atuar. Os dados que alimentam a BC estão descritos no Apêndice E.

5.3.1 Experimento 1

O objetivo desse experimento é demonstrar a capacidade do SCAE diagnosticar situações de anormalidade que estão além do conhecimento local da UP. Exemplo disso é o rompimento em uma linha de distribuição que conduz o óleo extraído do poço até a etapa de coleta. Nele são utilizadas duas UPs que, após a troca de informações sobre suas condições operacionais podem deduzir que a causa do problema está na linha que as interliga.

Para simular o problema, foi inserido no programa de controle da ETO uma variável para indicar o rompimento da linha, quando ela for ativada, o óleo enviado pela PCO não irá alterar o nível do tanque da ECO, dessa forma têm-se a situação de rompimento da linha.

Inicialmente, o agente de análise local (AAL) da ECO verifica os limites operacionais da variável vazão de entrada que representa o volume do fluido enviado pelas PCOs. Ao identificar que a mesma encontra-se fora dos limites de controle, inicia-se o processo de análise.

Primeiramente, o diagnóstico "Queda na vazão de entrada" é identificado (Figura 38 - instante 240). Como nenhuma ação local foi selecionada, o diagnóstico é enviado para as UPs.

O agente de sincronização (ASI) da PCO recebe o diagnóstico da ECO e o coloca na memória de trabalho, indicando ao AAL a existência do diagnóstico para que o mesmo inicie a análise.

Ao analisar as condições locais e, não havendo anormalidades no funcionamento da UP, e o diagnóstico enviado para ECO, ele conclui que o problema possivelmente está na linha que conduz o óleo, portanto, a ação local "Desligar UB" é selecionada e enviada ao agente de atualização do controlador (AAC) para execução.

O AAC verifica que a ação não gera impacto negativo em outras células e então envia o

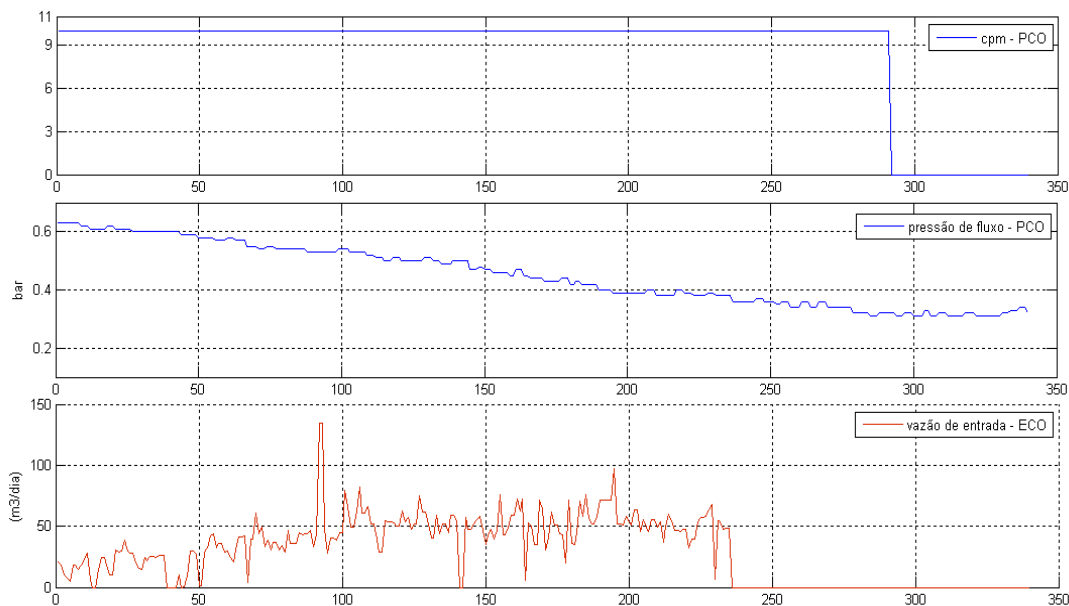



Figura 38: Dinâmica do processo

ajuste para o controlador. Assim, as PCOs tem o seu CPM ajustado para zero, ou seja, desliga a UB, evitando o prolongamento da situação (Figura 38 - instante 280).

Na Figura 39 é mostrado a sequência de mensagens trocadas pelos agentes durante a solução do problema.

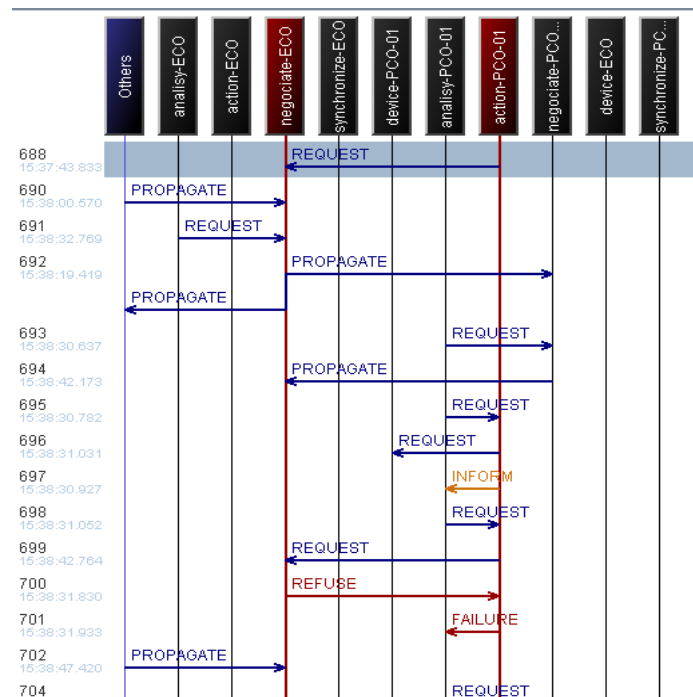



Figura 39: Sequência de mensagens dos agentes

5.3.2 Experimento 2

O segundo experimento visa a demonstrar o mecanismo de interação entre UPs, no caso, PCO e ECO, a fim de encontrar uma solução para uma condição de anormalidade em PCO. Para isso, é realizada uma negociação entre as UPs, para verificar o impacto do ajuste de PCO na ECO.

Inicialmente, o AAL da PCO verifica o limite de controle da variável “nível dinâmico” e, ao perceber que o mesmo encontra-se acima do limite desejado, inicia o processo de análise, onde os diagnósticos “Nível médio do anular” e “Baixa eficiência da bomba” são identificados.

Após a identificação dos diagnósticos, o agente continua buscando uma solução em sua base de regras, até que a ação “Aumentar cpm” é selecionada para corrigir as anormalidades.

Ele envia uma mensagem para o AAC que, por sua vez, analisa o impacto da ação em outras UPs, identificando que a mesma terá impacto no funcionamento da ECO.

O ajuste terá, portanto, que ser negociado antes com a ECO. Para isso, o AAC envia uma mensagem aos agentes de negociação (ANE) das UPs que serão afetadas pela mudança que, por sua

vez, respondem à negociação baseadas nas regras descritas anteriormente para esse processo. A dinâmica desse processo, desde a identificação da anormalidade até o ajuste final, pode ser vista na Figura 40.

A partir do instante 0 até o instante 600, o valor do cpm da PCO foi sendo ajustado para aumentar a eficiência da bomba e trazer o nível dinâmico para os limites estabelecidos de controle, ilustrados por CTR(max) e CTR(min). Como o nível do tanque de ECO (H) está dentro dos limites de controle, e ainda, até aquele instante, nenhuma anormalidade foi diagnosticada para a UP, o ajuste é autorizado.

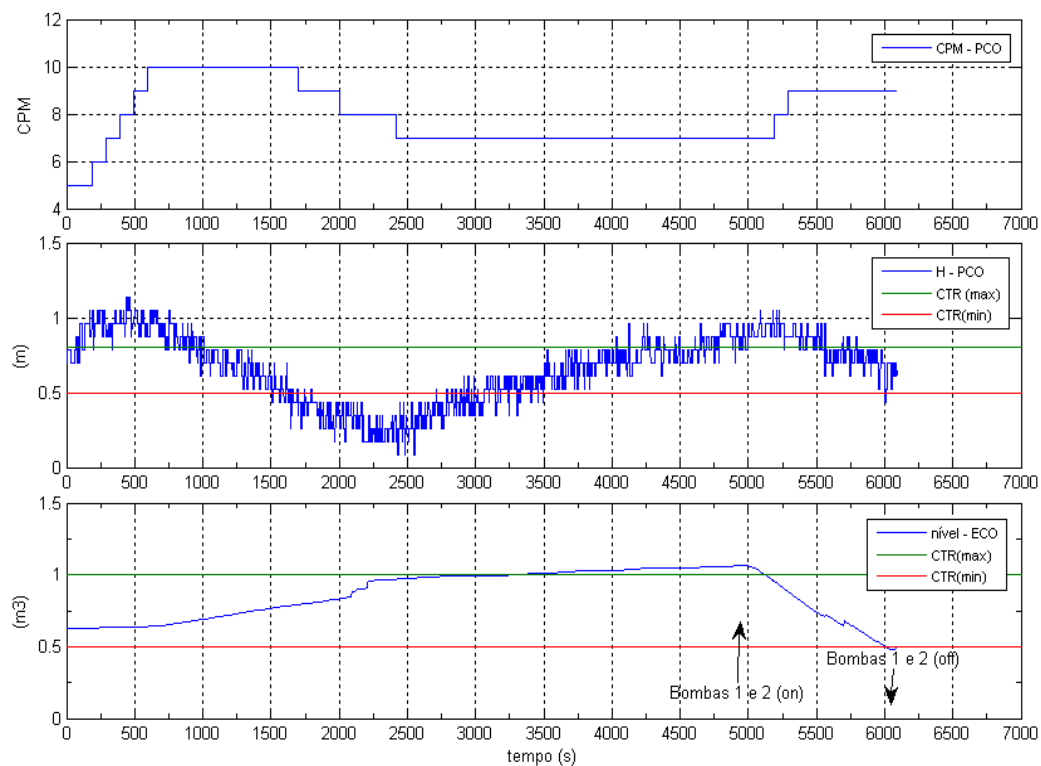



Figura 40: Dinâmica do processo

Na Figura 41 é visto o "diálogo" entre os agentes das UPs (PCO e ECO). É também possível ver a resposta "AGREE" (mensagens 150 e 162) confirmando que a ação pode ser executada sem prejuízos às UPs envolvidas. Dessa forma, o ajuste do cpm é enviado ao agente de verificação continuada (AVC) que escreve o novo valor no controlador. O processo é repetido até que a condição de anormalidade seja solucionada.

Já no instante 4300 novamente um alerta de variável fora do limite de controle é feito e o processo de análise iniciado. A ação "Aumentar cpm" é novamente identificada. Entretanto, a UP

ECO enfrenta problemas com o nível do tanque fora do limite de controle. Em decorrência, novos ajustes que possam aumentar a vazão de entrada e, consequentemente, o nível do tanque, não podem ser realizados até que a situação tenha sido corrigida. Dessa forma, o ajuste é negado.

Assim, PCO permanece sem poder ajustar o cpm, até que ECO tenha encontrado uma solução para o nível do tanque, o que acontece no instante 4900, quando as bombas são acionadas após a identificação das ações “Ligar Bomba 1” e “Ligar Bomba 2”. Em seguida, no instante 5200, já com o nível dentro dos limites desejados, a ECO autoriza a UP PCO a ajustar o cpm. O “diálogo” é ilustrado na Figura 42, onde é possível ver a mensagem “REFUSE” (mensagens 215, 225, 231 e 237) indicando a negação do ajuste.

Figura 41: Sequência de mensagens entre os agentes da PCO e ECO para “Ajuste liberado”

A partir do comportamento do sistema durante esse experimento, é possível verificar a noção de controle global do mesmo.

Da mesma forma, o paralelismo dos agentes permitiu que, mesmo enquanto a UP ECO está respondendo à negociação com a PCO, seus agentes locais estão atuando na tentativa de manter o adequado funcionamento da UP. Isso pode ser constatado no instante 4900 quando as bombas da ECO foram acionadas com o objetivo de ajustar o nível do tanque.

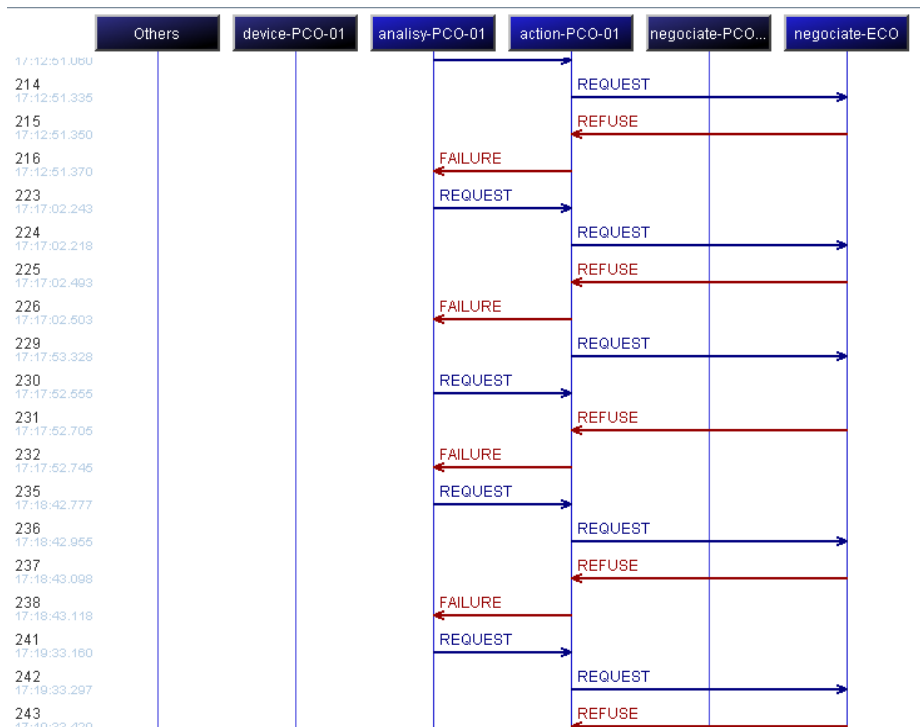



Figura 42: Sequência de mensagens entre os agentes PCO e ECO para "Ajuste negado"

5.3.3 Experimento 3

Neste experimento, buscou-se demonstrar, principalmente, a capacidade de resolução distribuída de problemas do sistema. Diferentemente dos anteriores, foram utilizadas três diferentes UPs: PCO, ECO e ETO. O experimento consiste em simular uma situação na qual a ETO apresente problemas para manter estabilizado o nível do tanque. Para isso, foram acrescentadas condições de falha nas bombas que enviam o óleo dos tanques da ETO para as etapas posteriores. Assim, a falha forçará uma ação por parte das outras UPs, com o objetivo de evitar, por exemplo, o transbordamento do tanque (Figura 43).

Inicialmente, o AAL da ETO verifica que o nível do tanque está fora do limite de controle. Após investigar sua base de regras, o diagnóstico “Nível alto do tanque” é identificado. Em seguida, a ação “Ligar bomba 1” é selecionada e enviada ao AAC, que verifica não ser uma ação com impacto nas demais UPs (Figura 43 - instante 1300). Dessa forma, o ajuste é enviado ao controlador pelo AVC. Em seguida, como o diagnóstico persiste, a segunda bomba é acionada através da ação

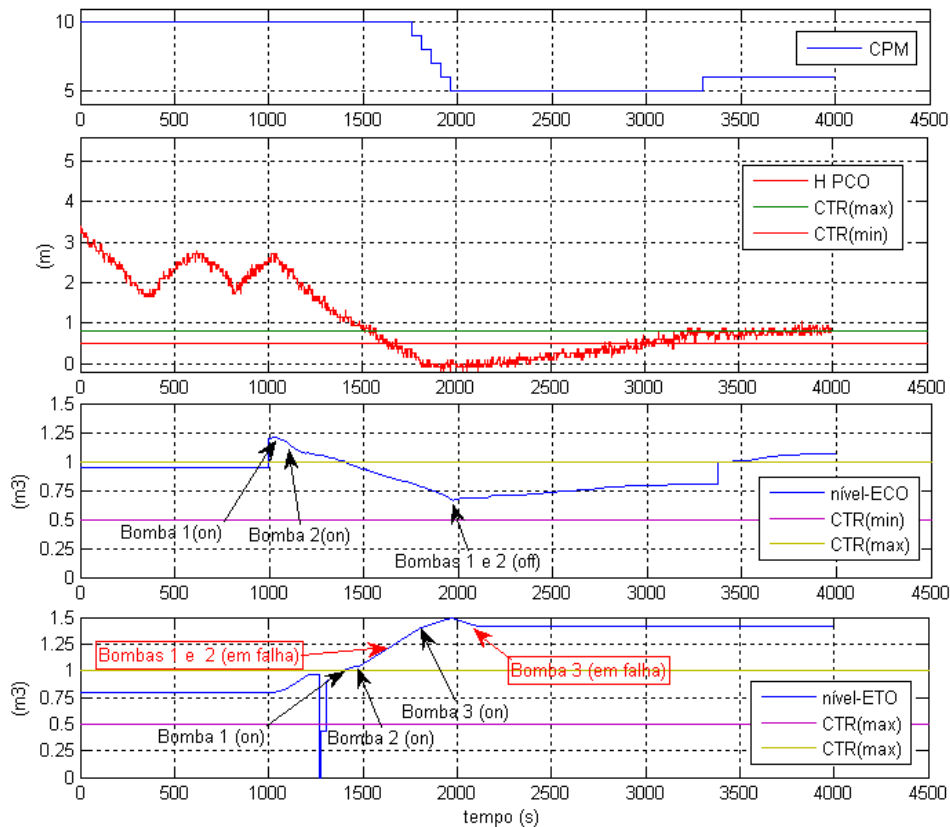



Figura 43: Dinâmica do processo

"Ligar bomba 2" (Figura 43 - instante 1500). A partir do instante 1600, as bombas 1 e 2 falharam. Persistindo o problema, a bomba 3 é acionada (instante 1800) pela ação "Ligar bomba 3".

Com as outras bombas em falha, o diagnóstico "Nível máximo do tanque" é identificado, não havendo mais nenhuma ação local para solucionar o problema, uma vez que todas as bombas foram acionadas e um diagnóstico de "Impossível esvaziar tanque" é identificado. Como em sua base de regras não existe nenhuma solução para o problema, o diagnóstico é propagado para a UP ECO, para que a mesma tenha "conhecimento" das condições operacionais de ETO e, assim, proceda aos ajustes necessários para evitar o agravamento do problema.

Ao receber o diagnóstico, a ECO identifica a necessidade de desativar suas bombas para reduzir o envio de fluido para a ETO, até que a mesma volte ao funcionamento normal, então executa as ações "Desligar bomba 1", "Desligar bomba 2" (Figura 43 - instante 2000).

Com o decorrer do tempo, e por conta do envio de fluido por parte da PCO, o nível da ECO continua aumentando (Figura 44) . Sem que nada possa ser feito localmente para reduzir o nível do tanque, a ECO identifica o diagnóstico de "Impossível esvaziar tanque" é o propaga para as UP relacionadas (PCO e ETO).

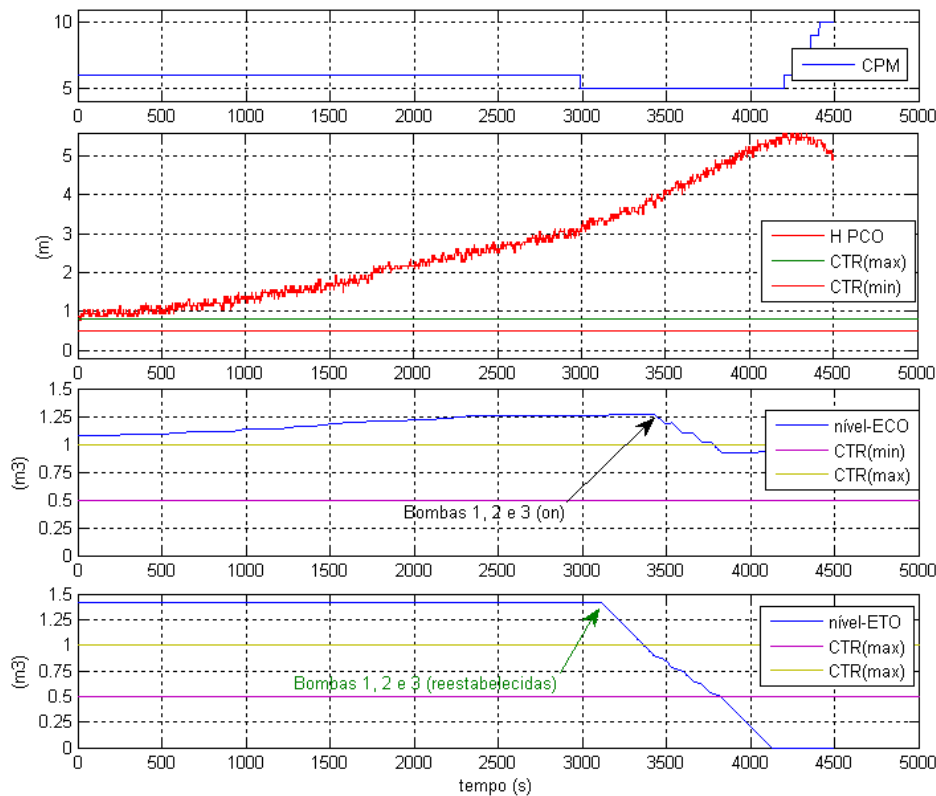



Figura 44: Dinâmica do processo

A PCO, ao receber o diagnóstico da ECO, faz uma análise das condições e regras locais que levam à seleção da ação “Reduzir cpm”, com o objetivo de reduzir o fluxo de óleo para a ECO, até que o nível seja restabelecido (Figura 44 - instante 2900).No instante 3200 (Figura 44), as bombas da ETO são restabelecidas e o processo volta à normalidade.

Esse experimento permite uma visão do comportamento cooperativo do sistema, uma vez que as demais unidades (PCO e ECO) ajustaram suas operações a fim de evitar o agravamento dos problemas enfrentados por ETO. A capacidade de comunicação, em nível de conhecimento, representada pela propagação dos diagnósticos, elimina a necessidade de troca de outros dados por parte das UPs.

Ainda o comportamento autônomo permitiu que, logo após o restabelecimento da capacidade de movimentação de fluxo da ETO, as outras UPs automaticamente tivessem suas condições operacionais ajustadas.

5.3.4 Avaliação do desempenho

A utilização da abordagem proposta no SCAE permitiu reduzir o tempo tomado para identificar uma condição anormal em uma UP, em relação à abordagem atual. Na Figura 45 é ilustrado o desempenho da análise com e sem a utilização do SCAE.

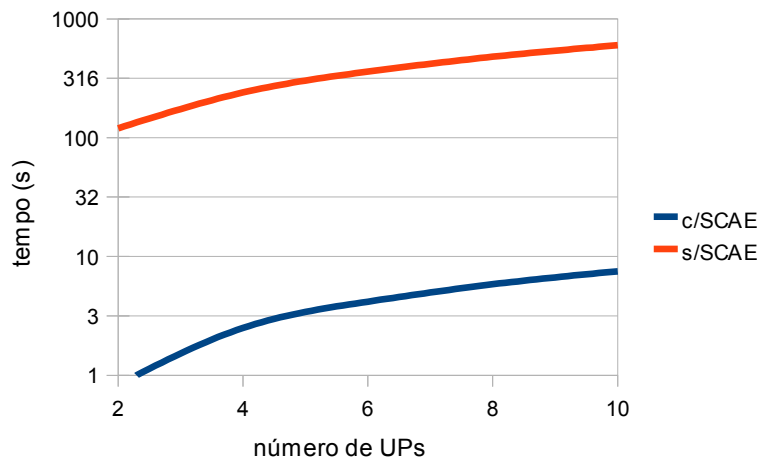



Figura 45: Desempenho do SCAE

Para essa avaliação, foi considerado apenas o tempo tomado para o envio e recebimento dos dados entre UPs e a COP. Também não foram considerados atrasos na rede ou falhas no envio.

Na estrutura convencional, onde cada UP utiliza um intervalo fixo de 60s para o envio de seus dados, para fins de comparação, o campo de produção de Buracica, da Petrobras UO-BA, usa uma janela de 180s na comunicação de cada controlador. Tem-se que, para que um diagnóstico envolvendo duas UPs seja realizado, o tempo mínimo será:

$$Ta = Nup.Ti$$

Onde, Ta é o tempo tomado para uma análise, Nup o número de UPs envolvidas e Ti o intervalo de tempo usado para enviar os dados a COP.

Já com a utilização do SCAE, considerando a mesma taxa de transmissão da rede utilizada no campo, que é de aproximadamente 2,4 kB/s, e uma média de 1,95 kB para cada mensagem trocada entre os agentes de duas UPs, obtida a partir das mensagens trocadas durante os experimentos, têm-se que o tempo tomado para a identificação de um diagnóstico envolvendo mais de uma UP seria:

$$Ta = \frac{m}{tx} Nup - 1$$

Onde, *m* é o tamanho da mensagem, *tx* é a taxa de transferência da rede.

Aumentando o número de UPs participantes, têm-se as curvas ilustradas na Figura 45.

Dessa forma, é possível ver que um dos maiores benefícios da utilização do SCAE é a redução no tempo de análise e atuação, em situações que afetam o resultado final da operação do campo.

6 CONSIDERAÇÕES FINAIS

Apesar das limitações e desafios encontrados, o presente trabalho demonstrou, de maneira experimental, como os conceitos de SMA e SD podem aumentar a eficiência dos sistemas de controle para campos de produção de petróleo, uma vez que podem atuar no nível de conhecimento identificando situações anormais, mesmo aquelas que dependem de um conhecimento global do sistema e que estão além da capacidade de visão local dos sistemas de controle.

Essa abordagem trás como uma das principais contribuições, enxergar o campo como unidades inter-relacionadas, dessa forma, as ações de controle sobre uma unidade podem levar em consideração dados ou as condições operacionais de outras unidades. Assim, é possível tratar localmente situações que anteriormente eram feitas apenas após a análise e intervenção de um operador.

O conceito de inteligência atribuído aos agentes por meio da cooperação entre eles pôde ser observado nos experimentos realizados, sobretudo no experimento 3, onde foi possível demonstrar como um mecanismo simples de propagar as condições operacionais de cada unidade para as demais trouxe a noção de inteligência e conhecimento global do sistema. Isso permitiu que, até certo limite imposto pelos requisitos de segurança, o campo pudesse funcionar de maneira autônoma, ajustando sua operação dinamicamente.

Por outro lado, o desenvolvimento desse trabalho permitiu observar que grande parte do esforço para a construção de um SMA está nas tarefas de modelar os agentes, seus comportamentos e a interação entre eles. Portanto, essa pesquisa reafirma a recomendação de, ao iniciar o desenvolvimento de um SMA, adotar uma metodologia que, além de ser amplamente documentada, possua ferramentas para auxiliar o projeto do sistema, e que seja baseada em métodos ágeis de desenvolvimento. Algumas metodologias para o desenvolvimento de SMA, tais como, PASSI

(Cossentino e Potts, 2002) que sugere o desenvolvimento incremental de tais sistemas, partindo dos requisitos iniciar até se chegar aos agentes e seus comportamentos propriamente ditos, refinamentos graduais são propostos a fim de alcançar a melhor forma para a distribuição das tarefas em um SMA, o que terá impacto sobretudo no nível de paralelismo em que o sistema irá operar, e consequentemente na sua velocidade para a identificação de situações e proposta de soluções.

Outra constatação importante dessa pesquisa foi em relação à plataforma JADE, que se mostrou uma ferramenta poderosa e eficaz para a implementação de SMA. Seus mecanismos de construção dos agentes, baseado na composição de comportamentos, permitem construir, com certa facilidade, agentes com estrutura de ações complexas e não determinísticas. Sabe-se que uma grande parte do esforço no desenvolvimento de SMA está na infraestrutura básica para funcionamento do mesmo, serviços como troca de mensagens, localização de agentes e manutenção do ciclo de vida dos agentes requerem conhecimento e tempo para a sua construção.

A utilização da plataforma JADE permite que o desenvolvedor de um SMA mantenha seu foco em atender aos requisitos específicos na aplicação, uma vez que ela oferece toda a infraestrutura necessária. Além de seus serviços serem bem testados e documentados conferindo maior confiabilidade ao sistema desenvolvido, sem que o programador concentre-se nesses aspectos.

6.1 CONTRIBUIÇÕES DA PESQUISA

Algumas das contribuições que podem ser percebidas a partir dos resultados desta pesquisa são:

1. Incentivo à utilização dos SMA nos sistemas industriais, sobretudo na atividade de produção de petróleo, na qual suas habilidades se encaixaram muito bem. A apesar de diversos trabalhos terem sido publicados nos últimos anos demonstrando o uso dos SMA na industria, alguns projetistas de sistemas de controle industrial argumentam que faltam demonstrações práticas sobre o uso da tecnologia, ao demonstrar o funcionamento em um ambiente simulado mas, com características de um campo real;
2. Construção de um modelo simplificado de um campo de produção de petróleo, o qual poderá servir como plataforma de teste para outras pesquisas. O modelo foi construído para integrar-se a estrutura laboratorial existente no LEA, dessa forma é

possível representar mais de uma etapa da produção de petróleo;

3. Integração do modelo ao laboratório do LEA, permitindo, assim, utilizar não apenas os módulos simulados, mas o SBM real instalado no LEA;
4. Construção de uma ferramenta computacional apta a desempenhar as tarefas de operação de um campo de produção de petróleo;
5. Apresentação da plataforma JADE para a construção de SMA. A experimentação prática em um ambiente heterogêneo como ocorre nos campos de produção, contribui para demonstrar como JADE pode ser aplicada em diversos ambientes, uma vez que suas interfaces permitem uma fácil integração da mesma com outras ferramentas e ou serviços de comunicação utilizados no ambiente industrial. A exemplo, o padrão de comunicação OPC.

Entretanto, acreditamos que as maiores contribuições surgirão ao longo do tempo, à medida que outras pesquisas possam seguir as trilhas iniciadas aqui assim como, esperamos que, no futuro o conceito do SCAE passa ser testado em ambiente de produção reais, servindo para assegurar aos projetistas de sistemas de controle industrial, a aderência do conceito multi-agentes em tais cenários.

6.2 LIMITAÇÕES DA PESQUISA

Alguns fatores, tais como restrições de tempo, problemas operacionais típicos da pesquisa experimental e de aprendizado no domínio da área de sistemas para manufatura, trouxeram algumas limitações para esta pesquisa. São elas:

- Não foram aprofundados os modelos matemáticos para construção do simulador, o que permitiria aproximar ainda mais das condições reais de funcionamento do campo;
- Não foram realizados casos de testes envolvendo uma maior quantidade de UPs, o que permitiria analisar mais a fundo a proposta de funcionamento colaborativo;
- Os testes realizados foram conduzidos com uma base de regras limitada, não ensejando, assim, que se contemplassem outras situações de anormalidade, além das utilizadas, e que poderiam ser úteis ao aperfeiçoamento da robustez do modelo;

- Não foram realizados testes de desempenho do sistema ante as condições reais de um campo de produção de petróleo, a fim de verificar, por exemplo, o efeito causado por atrasos na troca de mensagens, falhas em componentes do sistema ou falhas no canal de comunicação, situações propícias de ocorrerem no campo. Assim como, testes que verificassem o impacto da troca de mensagens entre os agentes na rede de comunicação e, como isso poderia afetar o desempenho geral dos sistemas de controle;
- O mecanismo de negociação desenvolvido permitia lidar apenas com três tipos de respostas (autorizado, negado ou em estabilização), não permitindo, por exemplo, que outras UPs, ao não concordarem com uma solicitação de ajuste, fizessem suas propostas de valores de acordo com os seus modelos localmente já ajustados.

6.3 SUGESTÕES PARA TRABALHOS FUTUROS

Para futuro desenvolvimento da pesquisa, sugere-se:

- Incorporar ao SCAE outros mecanismos para representação do conhecimento e raciocínio dos agentes, a exemplo de Lógica Nebulosa e Redes Neurais. O uso de tais mecanismos pode ajudar a tratar de forma mais eficiente as situações em que o conhecimento de especialistas é útil para lidar com incertezas do ambiente. Bem como, a escolha de ações poderia contar com uma representação que melhor se aproximasse do modo com as decisões são feitas pelo especialista, evitando desse modo ações de controle desnecessárias ou que não trouxessem ganhos significativos ao funcionamento do sistema;
- Desenvolver hardware para embarcar o SCAE como parte integrada do dispositivo de controle, ou seja, projetar controladores já aptos a operar segundo os conceitos e requisitos do SCAE;
- Realizar testes do sistema utilizando uma abordagem centralizada para a hospedagem das instâncias da PLA, que representam cada unidade da produção, desse modo, o SCAE poderia ser testado nos ambientes de supervisão e controle existentes nos campos da UO/BA, sem qualquer necessidade de alteração na infraestrutura computacional existente no campo.

- Realizar a modelagem matemática das unidades desenvolvidas no simulador, para que o mesmo possa operar de maneira mais próxima das condições reais, considerando a dinâmica dos fluídos, distâncias das unidades e obstáculos naturais;
- Especificar uma solução (hardware) para o canal de comunicação entre UPs, considerando o ambiente inóspito, as distâncias entre as UPs, a segurança da informação, os custos de implantação e manutenção. Tal infraestrutura de comunicação compreende um canal de comunicação sem fio para interligar as UPs do campo, e que representa um importante desafio para sua aplicação num ambiente de produção real, uma vez que a rede de comunicação disponível nos campos, preveem a interconexão apenas das UPs e a COP;
- Implementar o AIU, que deve realizar as tarefas do MGR do GCAD, e integrá-lo aos sistemas supervisórios, permitindo que os diagnósticos e/ou ações realizados pelo SCAE sejam armazenados em sistemas corporativos para outros fins e acompanhados por um operador. Assim como, o operador possa aplicar ações de controle nas UPs utilizando o SCAE e, dessa forma beneficiar-se dos mecanismos de negociação e sincronização oferecidos pelo mesmo, ou seja caberia ao sistema avaliar a possibilidade e os impactos de aplicação da ação proposta pelo operador;
- Avaliar o funcionamento do SCAE em outros ambientes, onde a estratégia de controle distribuído usando o modelo *Peer-to-Peer* possa ser aplicada. Apesar do desenvolvimento do SCAE manter o foco nos requisitos para operação de um campo de produção de petróleo, acreditamos que, o sistema desenvolvido pode ser aplicado a qualquer ambiente industrial no qual seja possível identificar o comportamento de células autônomas, quer seja em um ambiente geograficamente disperso, ou em sistemas industriais centralizados.

Acreditamentos que o amadurecimento dessa pesquisa, pode resultar em uma nova abordagem para o controle e gerenciamento de campos de produção de petróleo, e contribuir para o desenvolvimento da tecnologia de campos inteligentes, permitindo alcançar um cenário de controle completamente integrado e autônomo, não apenas no nível da produção do campo, mas, permitir ajustar essa produção as condições externas, tais como preço do barril, condições econômicas e recursos disponíveis.

REFERÊNCIAS

AL-ARNAOUT, I. H.; AL-DRIWEESH, S. M.; AL-ZARANI, R. M. *Intelligent Wells to Intelligent Fields: Remotely Operated Smart Well Completions.* SPE Intelligent Energy Conference, Amsterdam, Holanda, Fevereiro, 2008.

ALMEIDA, L. F. **Sistema Híbrido de Otimização de Estratégias de Controle de Válvulas de Poços Inteligentes sob Incertezas**. Tese de Doutorado, DEEPUC/RJ, 2007.

ARAÚJO, F. A. **Cálculo Hidráulico de uma Malha de Escoamento de Petróleo Utilizando Redes Neurais**. Dissertação (Mestrado). Universidade Federal do Rio de Janeiro, COPPE, 2007.

ARTERO, Almir O. **Inteligência Artificial: Teórica e Prática**. 1ª edição. São Paulo, Livraria da Física, 2008.

BARRETO FILHO, M. A. *et al*. **Proposta de um Sistema de Bombeio Mecânico em modelo reduzido para validação de modelos teóricos**. V Congresso Nacional de Engenharia Mecânica, Salvador, Bahia, 2008.

BELLIFEMINE, F.; CAIRE, G.; GREENWOOD, D. *Developing Multi-Agent Systems with JADE.* John Wiley & Sons, England, ISBN: 978-0-470-05747-6 (HB), 2007.

BEZERRA, M. V. **Avaliação de Métodos de Elevação Artificial de Petróleo Utilizando Conjuntos Nebulosos**. Dissertação (Mestrado). Campinas: Faculdade de Engenharia Mecânica e Instituto de Geociências, Universidade Estadual de Campinas, 2002.

BORDINI, R. H.; VIEIRA, R.; MOREIRA, A. F. **Fundamentos de sistemas multiagentes**. In: FERREIRA, C. E. (Ed.). Jornada de Atualização em Informática (JAI'01). Fortaleza, Brasil: SBC, 2001. v. 2, cap. 1, p. 3–44.

BRIOLA, D.; MASCARDI, V.; MARTELLI, M.; ARECCO, G.; CACCIA, R.; MILANI, *C. A Prolog-Based MAS for Railway Signalling Monitoring: Implementation and Experiments.* Anais do WOA'08, 2008.

BUSSMANN, S.; JENNINGS, N.R.; WOOLDRIDGE, M. *Multiagent Systems for Manufacturing Control: A Design Methodology.* Springer. In: A.H. Bond and L. Gasser, Editors, 2004.

COSSENTINO, M.; POTTS, C. *PASSI: a Process for Specifying and Implementing*

Multi-Agent Systems Using UML, 2002.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos: Conceitos e Projeto**, 4ª edição, Bookman, 2007.

FIPA – *The Foundation for Intelligent Physical Agents.* Disponível em: http://www.fipa.org, Acessado em: 01 de jan. 2010.

FREITAS, F.; BITTENCOURT, G. **Comunicação entre Agentes em Ambientes Distribuídos Abertos: o Modelo "peer-to-peer"**. Revista Eletrônica de Iniciação Científica (REIC). Ano II No. II Vol. II, Edição de Junho de 2002. Sociedade Brasileira de Computação (SBC). Brasil.

GAO, C.; RAJESWARAN, T. *A literature review on smart-well technology.* In procedings of SPE Production and Operations Symposium, Oklahoma City, Oklahoma, 2001.

GIRARDI, **R. Engenharia de Software baseada em Agentes**. Itajai, Santa Catarina, Brasil: Anais do IV Congresso Brasileiro de Ciência da Computação (CBCOMP 2004). Ed. Univali, 2004.

HECK, F.; LAENGLE, T.; WOERN, H. *A multi-agent based monitoring and diagnostics system for industrial components.* Proceedings of the DX´98, pp. 63-69, 1998.

HÜBNER, J. F; SICHMAN, J. **Organização de sistemas multiagentes**. In Proceedings of III Jornada de Mini-Cursos de Inteligência Artificial, Campinas, pp.247-296, 2003.

HÜBNER, J., BORDINI, R., Vieira, R. **Introdução ao desenvolvimento de sistemas multiagentes com Jason**. In XII Escola de Informática da SBC, volume 2, pág. 51–89, Guarapuava. UNICENTRO, 2004.

JADE – *Java Agent Development Enviroment.* Disponível em: http://jade.tilab.com/, Acessado em: 1 de jan. 2010.

JENNINGS, N.R. *An Agent-Based Approach for Building Complex Software Systems.* Comm. ACM, vol. 44, no. 4, pp. 35-41, 2001.

LI, X.; ZHANG, C.; GAO, L.; LI, W.; SHAO, X. *An agent-based approach for integrated process planning and scheduling*. Expert Systems with Applications, vol. 37, no. 2, pp. 1256–1264, Mar. 2010.

MARTINEZ, J. K.; KONOPCZYNSKI, M. R. *Integrated Reservoir Management in an Intelligent Well Environment.* SPE 77853, Melbourne, Australia, 2002.

MATURANA, F., P.; TICHÝ, P.; SLECHTA, P.; DISCENZO, F.; STARON, R., J.; HALL, K. *Distributed multi-agent architecture for automation systems.* Expert Systems with Applications 26 (2004) (1), pp. 49–56.

MONOSTORI, L.; VÁNCZA, J.;KUMARA, S. R. T. *Agent-based Systems for Manufacturing.* CIRP 52 (2), 2006, pp. 697–721.

MORENO, A.; VALLS, A.; VIEJO, A. *Using JADE-LEAP to implement agents in mobile devices.* In: TILAB EXP in search of innovation, Italy (2003), Disponível em: http://jade.tilab.com/papers-

exp.ht, Acessado em: 18 de mai 2010.

MOULIN, B., CHAIB-DRAA, B. *An overview of distributed artificial intelligence.* Foundations of Distributed Artificial Intelligence, 1996, pp3-55.

ORDOÑEZ, B. **Proposta de controle de operação de poços com Bombeio Mecânico através da pressão de fundo**. Dissertação (Mestrado). Universidade Federal de Santa Catarina, 2008.

PACHECO, L. A. **GCAD: Um Modelo Conceitual para Gerenciamento e Controle Autônomo e Distribuído para Sistemas Industriais Automatizados**. Dissertação (Mestrado), Universidade Federal da Bahia, 2011.

PACHECO, L. A.; LEPIKSON, H. A. *An Autonomous Control Strategy Alternative for Critical Industrial Automated Systems.* In: INCOM 2009 – Information Control Problems in Manufacturing, Moscow. IFAC – International Federation of Automatic Control. DOI: 10.3182/20090603-3-RU-2001.00054, 2009.

PATRÍCIO, A. R. **Estudo de um Sistema Inteligente para Elevação de Poços e Controle de Processos Petrolíferos**. Tese (Doutorado). Unicamp, 1996.

SAPUTELLI, L.; NIKOLAOU, M.; ECONOMIDES, M. J. *Self-Learning Reservoir Management.* SPE 84064, 2005.

SHEN, W.; HAO, Q.; YOON, H. J.; Norrie, D.H. *Applications of agent-based systems in intelligent manufacturing: an updated review.* Advanced Engineering Informatics 20, 2006, pp. 415–431.

SILVA, C. T. L. **Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropos**. Dissertação (Mestrado) – Universidade Federal de Pernambuco, 182f , 2003.

SILVA, L. C. F. **Inteligência Computacional para Predição de Produção de Reservatórios de Petróleo.** Tese de Doutorado, COPPE/UFRJ, 2006.

STEEN, E. van der. *An Evolution From Smart Wells to Smart Fields.* Digital Energy Conference and Exhibition, Houston, Texas, U.S.A, 2006.

SYCARA, K. P. *Multiagent Systems.* AI Magazine 19(2): 79-92, 1998.

TANENBAUM, A. S.; STEEN, M. V. **Sistemas Distribuídos**, 3ª ed, Rio de Janeiro, Prentice Hall Brasil, 2008.

THOMAS, J. E. (org.) **Fundamentos de Engenharia de Petróleo**. RJ: Ed. Interciência, 2004, ISBN 85-7193-099-6.

TVEIT, A. *A Survey of Agent-Oriented Software Engineering.* NTNU Comp. Sci. Graduate Student Conf., 2001.

TWEEDALE, J.; ICHALKARANJE, N.; SIOUTIS, C.; JARVIS, B.; CONSOLI, A.; PHILLIPS-WREN, G. *Innovations in multi-agent systems.* J. Netw. Comput. Appl., vol. 30, 2007, pp. 1089.

VRBA, K.; HALL, H.; MATURANA, F. P. *Rockwell automation agents for manufacturing.* in Proc. 4th Int. Conf. Auton. Agents Multi Agent Syst. (AAMAS 2005), Jul. 25–29, 2005, pp. 147–153.

WAGNER, T. *An agent-oriented approach to industrial automation systems*. Technical report, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2002.

APÊNDICES

APÊNDICE A – AMBIENTE DE DESENVOLVIMENTO PARA SISTEMAS MULTIAGENTES

A ferramenta JADE (*Java Agent Development Enviroment*) provê toda a infraestrutura para explorar o paradigma de agentes na construção de aplicações distribuídas. Desenvolvido inicialmente pela Telecom Itália, o principal objetivo do JADE era prover uma infraestrutura independente para o gerenciamento do ciclo de vida dos agentes, e a comunicação entre eles. Por isso, sua implementação foi feita em Java, que é uma das linguagens de programação orientadas a objeto mais utilizada exatamente por oferecer uma arquitetura de execução independente de plataforma (JADE, 2010).

Para permitir que agentes rodando na plataforma JADE possam comunicar-se com agentes desenvolvidos em outras plataformas, foi adotado o padrão FIPA para a camada de comunicação, de forma que qualquer ambiente de execução de agentes que utiliza o FIPA possa interagir com os agentes desenvolvidos para a plataforma JADE.

Arquitetura

Segundo Bellifemine, Caire e Greenwood (2007), JADE é organizado através de um elemento no topo da arquitetura que é a plataforma. A plataforma JADE é formada por contêineres, onde os agentes residem. Um contêiner JADE é um processo Java rodando em uma máquina virtual. Os contêineres podem ser distribuídos sobre uma rede e integrados através do contêiner principal denominado main container (Figura 22).

O agente, por sua vez, é um objeto que reside em um contêiner, que oferece os serviços necessários para sua execução, bem como comunicação com os demais agentes (Figura 23).

Essa arquitetura permite que os agentes sejam dispersos ao longo da rede, mas possam se comunicar com outros agentes como se estivessem em um mesmo local. Para isso, cada agente registrado em uma plataforma receberá um identificador único de agente (BELLIFEMINE, CAIRE E GREENWOOD, 2007).

Estrutura do agente

A estrutura interna do agente JADE é baseada no modelo de agentes reativos, e é composta de (GIRARDI, 2004):

Comportamentos – formam as ações que cada agente realiza. Cada ação é implementada por meio de um ou mais comportamentos;

Caixa de mensagens – é uma estrutura de dados que armazena todas as mensagens recebidas pelo agente;

Escalonador de comportamento – é responsável por determinar a ordem de execução dos comportamentos;

Gerenciador do ciclo de vida – é responsável por controlar o estado atual do agente;

Recursos da aplicação – representam o conhecimento adquirido pelo agente durante a execução da aplicação;

Um agente JADE é multitarefa, ou seja, mais de um comportamento pode ser executado ao mesmo tempo.

O JADE não possui um mecanismo interno para representação e raciocínio sobre conhecimento, entretanto, possui componentes que permitem integrar outros mecanismos externos ao JADE (GIRARDI, 2004).

Executando agentes JADE em dispositivos embarcados

O LEAP (Lightweight Extensible Agent Platform) foi lançado em 1999, através de uma iniciativa de fabricantes de dispositivos móveis, como Motorola, Siemens e Ericsson, que desejavam construir uma plataforma para a execução de aplicações SMA que fosse suficientemente “leve”, para permitir sua execução em dispositivos móveis, e integrado à plataforma JADE. Com esse recurso, JADE passou a oferecer uma alternativa para a execução de agentes em dispositivos com recursos limitados, tais como: PDA e *smart devices* (BELLIFEMINE, CAIRE E GREENWOOD, 2007).

O LEAP oferece todas as funcionalidades da plataforma original, a exceção de algumas ferramentas de gerenciamento que, por utilizarem recursos de interface gráficas, não poderiam ser

executados satisfatoriamente em ambientes limitados. Portanto, JADE pode ser implantado em qualquer hardware onde uma máquina virtual Java possa ser implantada (MORENO, VALLS e VIEJO, 2003).

Com a utilização da especificação J2ME, uma máquina virtual Java pode ser incorporada em diversos dispositivos, que vão desde dispositivos com capacidades de processamento e armazenamento mais elevados até dispositivos de recursos extremamente limitados, como *smart cards*.

Como o plugin LEAP altera apenas o ambiente de execução da plataforma, um sistema pode ser implementado em JADE sem que seja necessário decidir, nesse momento, se o ambiente de execução será na plataforma original ou no LEAP.

O LEAP possibilitou um dos objetivos do sistema proposto, que era embarcar o software em um hardware de baixo custo, e que pudesse ser implantado junto aos dispositivos de campo da planta controlada.

APÊNDICE B – REPRESENTAÇÃO DO CONHECIMENTO

No contexto deste trabalho, conhecimento, segundo Artero (2009), é o armazenamento de informações e o uso de modelos por parte de uma pessoa ou máquina, com o objetivo de interpretar, identificar, predizer e responder ao mundo exterior. O uso por diferentes agentes (humanos ou computador) requer diferentes formas de representação.

Diversos modelos são apresentados na literatura para a representação de conhecimento em computador, e a escolha de um modelo de representação depende também do tipo de informação que será armazenada. Os tipos mais conhecidos de representação do conhecimento em computador são: lógica, regras de produção, redes semânticas, quadros e roteiros e a representação por árvore (ARTERO, 2009).

A representação por lógica evoluiu, ao longo dos anos a partir dos fundamentos criados por Aristóteles. A Lógica de Predicados e a Lógica Nebulosa são dois exemplos de representações que surgiram a partir da lógica proposta por Aristóteles (ARTERO, 2009).

Um importante exemplo da lógica de predicados é a linguagem Prolog, utilizada neste trabalho com o objetivo de representar o conhecimento do especialista sobre o ambiente onde o sistema irá atuar, de maneira a permitir aos agentes raciocinarem sobre fatos ocorridos no ambiente, de forma semelhante a que um ser humano faria.

Prolog é uma linguagem baseada na lógica de predicados. Sua escrita segue um formato declarativo, que facilita a construção de programas lógicos. Diferentes de outras linguagens de programação, no Prolog não são utilizadas estruturas lógicas de controle, que dizem ao computador o que ele deve fazer e como deve fazer. Em Prolog, são fornecidos ao computador fatos e regras. A partir daí, o mesmo pode, utilizando o motor de inferência do Prolog, gerar soluções para as questões que lhe são formuladas ou, ainda, deduzir novos fatos (ARTERO, 2009).

A máquina de inferência do Prolog utiliza a técnica de encadeamento para trás, que significa que uma conclusão é tida inicialmente como verdadeira para então buscar em uma base de regras e fatos algo que a comprove. Este procedimento é denominado Prova Indireta (ARTERO, 2009).

Apesar de existirem diversos comandos, não é necessário conhecer muitos deles, pois é

possível escrever um programa inteiro utilizando as próprias declarações do programador e o comando “se”, que é representado pelo símbolo “:-”. O código a seguir apresenta um programa simples capaz de determinar se um aluno está ou não aprovado.

```
nota(pedro,5,7).
nota(ana,4,3).
nota(luis,2,7).
nota(maria,8,7).
aprovado(X):-nota(X,N1,N2), Media is (N1+N2)/2, Media >= 5.
```

No referido código podem ser vistos alguns importantes elementos da programação em Prolog, tais como:

i) As variáveis `pedro,ana,luis` e `maria`.

ii) Os fatos `nota(pedro,5,7)`, `nota(ana,4,3)`, `nota(luis,2,7)` e `nota(maria,8,7)`.

iii) E a regra `aprovado(X):-nota(X,N1,N2), Media is (N1 + N2)/2, Media >= 5`.

A simplicidade das declarações em Prolog e o modelo de inferência utilizado permitem a construção de versões suficientemente leves do mecanismo de inferência, o que permite sua execução em equipamentos com recursos limitados, a exemplo, da biblioteca TuProlog (DENTI, OMICINI e RICCI, 2001) que é um motor de inferência para a linguagem Prolog implementado em linguagem Java utilizando a especificação J2ME[10], o que permite sua execução em dispositivos como PDAs e *smart phones*.

Baseado na natureza dirigida a regras dos sistemas de diagnóstico (BRIOLA *et al*., 2008), na simplicidade e flexibilidade de construção do conhecimento utilizando Prolog e, ainda, a possibilidade de execução de seus programas em dispositivos com recursos limitados garantida pela biblioteca TuProlog, ela foi adotada como mecanismo de raciocínio utilizado pelos agentes no sistema desenvolvido.

---

10 J2ME – Especificação de máquina virtual Java para execução em dispositivos com recursos limitados.

APÊNDICE C – FUNCIONAMENTO BÁSICO DE UMA UB

O funcionamento básico de uma UB ocorre conforme descrito a seguir (BEZERRA, 2002):

1. O acionamento do sistema BM é feito normalmente por um motor elétrico operando com uma velocidade que pode variar entre 500 e 1500 rpm. Na ausência de energia elétrica, pode-se utilizar um motor de combustão interna. Em sistemas automatizados, variadores de frequência permitem a regulagem dinâmica da rotação do motor;
2. O motor conecta-se ao redutor, que reduz a velocidade e suporta o torque de bombeio;
3. A UB possui um sistema de biela-manivela, que converte movimento de rotação em movimento alternativo, e que é transmitido até às hastes através do balancim (viga oscilante). Um contrapeso ajustável regula a carga imposta ao motor;
4. A haste polida é ligada ao balancim, e a selagem do equipamento é garantida através do sistema de vedação localizado acima do "T" de bombeio, na cabeça do poço, mantendo os fluidos dentro do poço;
5. A coluna de hastes conecta-se à haste polida e, dentro do poço, transmite o movimento alternado à bomba de fundo.

As cargas aplicadas sobre o motor e demais componentes da UB são especificadas, pressupondo-se que a UB esteja adequadamente montada sobre uma base perfeitamente horizontal e estável. A perda de integridade da base tem implicações sobre todos os componentes da UB, que estarão sujeitos a vibrações não previstas em seu dimensionamento e que poderão levá-los a falhas. Devido à grande quantidade de partes móveis e de mancais, é fundamental garantir que sua lubrificação ocorra de maneira adequada (THOMAS, 2004).

Na atualidade, o uso de sistemas de monitoração remota já trouxe bastante progresso para os sistemas de produção de petróleo, utilizando BM, através da leitura das cartas dinamométricas de superfície (CDS) e do registro do nível dinâmico (principais ferramentas disponíveis para avaliar as condições de funcionamento do BM) (BEZERRA, 2002).

A automatização de poços está caracterizada pela utilização de instrumentos como sensores e atuadores, que têm o objetivo de monitorar algumas variáveis de interesse referentes ao processo. Na elevação por bombeio mecânico, pelo menos duas medições estão disponíveis mediante o uso de sensores: posição e carga referidas à haste polida. Estas são medidas na superfície e correspondem aos dados fornecidos para a geração da CDS. A carta de fundo é calculada a partir desses dados fornecidos na superfície, utilizando modelos para o movimento dinâmico da coluna de hastes (ORDONEZ, 2008).

Para o monitoramento do sistema de bombeio, é importante contar com sensores de fundo, principalmente sobre a pressão no fundo do poço, já que a propagação de efeitos até a superfície pela coluna de hastes pode sofrer alterações significativas (ORDONEZ, 2008).

Em poços que operam com o BM, as CDS e CDF, que efetivamente auxiliam no controle, são utilizadas como principais ferramentas de controle e monitoramento. Dado seu histórico de uso, admite-se que elas representam um correto diagnóstico sobre o funcionamento dos componentes da bomba de fundo e, portanto, acerca do desempenho de bombeio (ORDONEZ, 2008).

Um dos principais problemas de eficiência é o enchimento incompleto da bomba. Isso é causado quando a capacidade da bomba excede a produção do reservatório, ou também existe uma pobre separação de gás na entrada da bomba e, assim, perde-se um pouco de deslocamento útil do pistão, devido à interferência do gás (ORDONEZ, 2008).

O primeiro passo é minimizar qualquer interferência de gás na bomba e livrar a bomba de problemas mecânicos. O passo seguinte é otimizar o deslocamento da bomba para remover todo fluido apto para elevação disponível no poço (ORDONEZ, 2008).

A velocidade da UB é medida em cpm e, com a instalação de um variador de frequência na superfície junto a UB, pode ser ajustada continuamente, variando a vazão de produção de forma suave, através de um controle VSD (Variable Speed Drive), e não de forma discreta, como ocorre utilizando-se o controle pump-off (bombeamento total ou nulo) (ORDONEZ, 2008).

Monitorando-se o nível de fluido no anular (estimado através da pressão de fundo) juntamente com o enchimento da câmara do pistão (através da CDF), é possível evitar a pancada de fluido (importante fenômeno que pode acentuar o desgaste dos componentes da bomba de fundo), através do controle em tempo real da velocidade de bombeio da UB (ORDONEZ, 2008).

O controle VSD está baseado em uma velocidade variável de bombeio propiciada utilizando um variador de frequência, permitindo com isso uma maior flexibilidade ao projeto, uma vez que, para manter o nível desejado no anular do poço de produção, pode-se ajustar a velocidade de

bombeio e, consequentemente, a vazão de óleo, conforme a necessidade de controle.

Durante de elevação por BM, deve-se assegurar o enchimento completo da bomba, visando um máximo de produção e evitando os problemas causados pelo seu enchimento parcial, tais como a pancada de fluido e os desgastes nos componentes da bomba (ORDONEZ, 2008).

Outro fator importante no desempenho da bomba está associado ao ponto de operação relacionado ao nível de fluido no anular. Este ponto de operação é caracterizado pelo enchimento completo da bomba com a menor pressão de fundo possível, o que proporciona uma menor contrapressão sobre os canhoneados do reservatório e incrementa a vazão de formação. O nível associado a esse ponto é próximo à sucção da bomba (ORDONEZ, 2008).

APÊNDICE D – OUTRAS DEFINIÇÕES DO GCAD

Requisitos atendidos pelo GCAD

Conforme Pacheco (2011), o GCAD pressupõe o atendimento dos seguintes requisitos principais:

Requisito 1. Capacidade de comunicação e decisão distribuída em rede

Caso seja detectada a necessidade de ajuste por parte de um MC em uma variável local que influencie na composição de um valor global do processo, esta atualização deverá seguir um processo de:

-Negociação, de modo que cada célula relacionada analise seu cenário operacional com base no ajuste proposto por uma célula e então autorize ou informe restrições para o ajuste;

-Sincronização, de forma que a célula proponente notifique as células relacionadas quanto a ajustes locais emergenciais já realizados. Os ajustes sujeitos a sincronização são realizados localmente pela célula proponente e só depois são propagados às demais células envolvidas.

Requisito 2. Capacidade de leitura e escrita em MR

Cada MC deve estar habilitado a coletar, em tempo real, informações da MR, bem como ajustar seus parâmetros ou comandar ações de controle, sem necessariamente interromper o funcionamento local.

Requisito 3. Capacidade de armazenamento local

Um requisito importante é a capacidade de armazenamento local dos dados, de processo e de configuração, incluindo séries históricas de cada variável de interesse, no período de tempo definido. Estas séries, juntamente com a BC local, devem servir de insumo para o MC local realizar análises corretivas ou preventivas de forma autônoma. A capacidade local de armazenamento deve ser configurável, de acordo com a necessidade e importância estratégica de cada célula.

Requisito 4. Armazenamento em servidores de historiamento

Além do armazenamento local, havendo estrutura de rede disponível para tal, os dados de processo, de configuração e regras de conhecimento devem ser continuamente enviados a um servidor remoto (o MGR), de forma a permitir o historiamento de dados, bem como futuras análises para fins de simulação e otimização.

Requisito 5. Capacidade de sincronização de dados, após a restauração do link de comunicação, em caso de perda de conexão

Ao se restabelecer o link de comunicação entre uma célula e o MGR, os dados armazenados localmente devem ser sincronizados, para que o MGR mantenha um registro completo dos dados de cada célula. Os arquivos devem ser automaticamente coletados pelo MGR e importados para os historiadores de dados e para a BC global, de forma a restaurar o histórico de dados e atualizações locais ocorridas durante a desconexão.

Requisito 6. Capacidade de funcionamento autônomo

Em caso de perda de conexão entre o MGR e uma célula, inclusive por falha no próprio MGR, as células devem ser capazes de funcionar em modo autônomo, tomando decisões, em conjunto com células relacionadas, sobre ajustes, ações e correções a serem realizados no sistema local. Além disso, a malha de controle que contém o MC deverá permanecer estável, mesmo com células inativas. Para as células ligadas a este MGR a desconexão deve ser inócua e, desde que sejam observadas as restrições locais de funcionamento e as restrições relacionadas a demais células, as operações locais devem permanecer ativas.

Requisito 7. Capacidade de decisão local, mesmo diante das incertezas do processo (ambiguidades)

Em caso de situações para as quais o sistema não disponha de uma regra ou ação precisa, é necessário que o MC tenha a habilidade de analisar as informações existentes (cenários de operação local, séries históricas, informações de demais células que possuam alguma relação ou semelhança à ocorrência observada) e inferir sobre que ação ou ajuste realizar na célula local. Em caso de conflitos ou incoerência na lógica entre os MC, o sistema local deve submeter a ação a níveis remotos ou a ações manuais, sinalizando o impasse e conduzindo o sistema a um modo seguro de operação.

Estrutura da base de conhecimento (BC)

As bases de conhecimento constituem um importante mecanismo para alcançar a autonomia e comportamento inteligente do sistema. Na proposta do GCAD, cada célula autônoma possui uma BC local, onde são registrados:

i)Dados do processo, que são as séries históricas relativas às variáveis de processo consideradas relevantes;

ii)Regras de conhecimento local;

iii)Dados globais;

iv)Interfaces com as demais células autônomas relacionadas, onde são armazenadas as informações referentes a relação entre elas.

Cada BC deve ser inicialmente alimentada pelo especialista, informando: variáveis e seus limites operacionais, relação entre variáveis de células distintas, ações corretivas e regras iniciais de conhecimento.

Os limites operacionais das variáveis constituem um importante mecanismo para o funcionamento do MC. Eles são utilizados para identificar quando uma variável deixa uma faixa de valores prevista para sua operação normal e quando uma análise baseada nas regras de conhecimento deve ser realizada, a fim de encontrar ações que visem corrigir o desvio.

Módulo de Gerenciamento Remoto (MGR)

A idéia principal do MGR é englobar tarefas de supervisão e adicionar tarefas de controle em nível remoto. Portanto, o MGR exerce o papel de um MC, porém com uma visão do processo como um todo. Realiza análises na BC global e em dados do historiador de processos.

Possui, além das capacidades de análise de informações baseadas em conhecimento e interação com outras células, mecanismos tais como: treinamento e aprendizagem, predição de falhas, simulação e ajustes, além de atualizar as BC locais das células sobre sua supervisão.

Para a tarefa de supervisão global da planta, a proposta do MGR prevê a distribuição das informações coletadas nas células para os especialistas através de uma IHM, que também deve permitir aos especialistas a realização de ajustes remotos nas células autônomas (PACHECO, 2009)

APÊNDICE E – MONTAGEM DAS BASES DE CONHECIMENTO

Base de conhecimento da UP PCO

a) Especificações da UP

```
//UP REMOTA (ECO)
RemoteCell cell = new RemoteCell("","ECO"); //NOME DA CÉLULA REMOTA
cell.setLocalVariable("VAZAO_BOMBA"); //NOME DA VARIÁVEL DE LIGAÇÃO (LOCAL)
cell.setRemoteVariable("NIVEL_TQ1"); //NOME DA VARIÁVEL DE LIGAÇÃO (REMOTA)
cell.setTypeOfRelationship(RemoteCell.UNDEFINED); //TIPO DE RELACIONAMENTO ENTRE
AS VARIÁVEIS
cell.setDependenceType(RemoteCell.INFLUENCE); //TIPO DE RELACIONAMENTO ENTRE AS
CÉLULAS
DBManager.getInstance().save(cell);

//VARIÁVEL REMOTA (NIVEL_TQ1)
RemoteVariable remVar = new RemoteVariable("NIVEL_TQ1", cell); //NOME DA
VARIÁVEL REMOTA
DBManager.getInstance().save(remVar);

//VARIÁVEL LOCAL (NIVEL_DINAMICO)
ProcessVariable var = new ProcessVariable("NIVEL_DINAMICO");
var.setTagName("[LEA]AAg.Nivel_h"); //TAG DO CONTROLADOR
var.setLimitCtMax(new RefValueDouble(new Double(0.8))); //LIM MÁXIMO
var.setLimitCtMin(new RefValueDouble(new Double(0.5))); //LIM MÍNIMO
var.setPersist(true); //MANTER HISTÓRICO DOS VALORES
DBManager.getInstance().save(var);

//VARIÁVEL LOCAL (PRESSAO_FLUXO)
var = new ProcessVariable("PRESSAO_FLUXO");
var.setTagName("[LEA]Program:MainProgram.PT_003"); //TAG DO CONTROLADOR
var.setPersist(true); //MANTER HISTORICO DOS VALORES
DBManager.getInstance().save(var);

ParameterVariable pvar = new ParameterVariable("VAZAO_BOMBA");
pvar.setDoubleValue(new Double(0)); //VALOR PADRÃO
DBManager.getInstance().save(pvar);

//VARIÁVEL LOCAL (CPM)
var = new ProcessVariable("CPM");
var.setTagName("[LEA]speed_data_use[1]"); //TAG DO CONTROLADOR
var.setDoubleValue(new Double(10.0)); //VALOR PADRÃO
var.setLimitCtMax(new RefValueDouble(new Double(11))); //LIM MÁXIMO
var.setLimitCtMin(new RefValueDouble(new Double(8))); //LIM MÍNIMO
var.setPersist(true); //MANTER HISTÓRICO DOS VALORES
DBManager.getInstance().save(var);

//AÇÕES
```

```
//AUMENTAR_CPM
AdjustAction act = new AdjustAction("AUMENTAR_CPM");
act.setGlobalInfluence(true); //POSSUI INFLUÊNCIA GLOBAL
act.setGlobalVariable("VAZAO_BOMBA"); //VARIÁVEL INFLUÊNCIADA
act.setAdjustType(AdjustAction.AT_INCREASE); //TIPO DE AJUSTE
act.setNewValue(new RefValueAbs(new Double(1),"CPM")); //NOVO VALOR
act.setVariableName("CPM"); //VARIÁVEL AJUSTADA
act.setDeadline(20000); //PRAZO DE EXECUÇÃO
DBManager.getInstance().save(act);

//REDUZIR_CPM
act = new AdjustAction("REDUZIR_CPM");
act.setGlobalInfluence(false); //POSSUI INFLUÊNCIA GLOBAL
act.setNewValue(new RefValueAbs(new Double(-1),"CPM")); //NOVO VALOR
act.setVariableName("CPM"); //VARIÁVEL AJUSTADA
act.setDeadline(20000); //PRAZO DE EXECUÇÃO
DBManager.getInstance().save(act);

//REDUZIR_CPM
act = new AdjustAction("DESLIGAR_UB");
act.setGlobalInfluence(false); //POSSUI INFLUÊNCIA GLOBAL
act.setNewValue(new RefValueFator(new Double(-0.05),"CPM")); //NOVO VALOR
act.setVariableName("CPM"); //VARIÁVEL AJUSTADA
act.setDeadline(20000); //PRAZO DE EXECUÇÃO
DBManager.getInstance().save(act);
```

b) Regras

```
diagnostico('NIVEL_MEDIO_ANULAR'):-variavel('NIVEL_DINAMICO',V), V>0.8, V<1.2.
diagnostico('NIVEL_ALTO_ANULAR'):-variavel('NIVEL_DINAMICO',V), V>=1.2.
diagnostico('NIVEL_BAIXO_ANULAR'):-variavel('NIVEL_DINAMICO',V), V<0.5.
diagnostico('BAIXA_EFICIENCIA'):-diagnostico('NIVEL_ALTO_ANULAR'),
variavel('CPM',V), V<10.0.

diagnostico('PROVAVEL_PANCADA_FLUIDO'):-
diagnostico('NIVEL_BAIXO_ANULAR'),variavel('PRESSAO_FLUXO',V),V<0.03.

diagnostico('PROVAVEL_FURO_COLUNA_TUBOS'):-diagnostico('NIVEL_ALTO_ANULAR'),
diagnostico('ECO/NIVEL_BAIXO_TANQUE'),
variavel('VAZAO_BOMBA',V), V > 2.0.
diagnostico('PROVAVEL_FURO_LINHA'):-
diagnostico('NIVEL_BAIXO_ANULAR'),diagnostico('ECO/QUEDA_VAZAO_ENTRADA').

acao('AUMENTAR_CPM'):-diagnostico('BAIXA_EFICIENCIA').
acao('AUMENTAR_CPM'):-diagnostico('NIVEL_MEDIO_ANULAR'),
variavel('CPM',V),V<10.0.
%acao('DESLIGAR_UB'):-diagnostico('PROVAVEL_FURO_COLUNA_TUBOS').
%acao('DESLIGAR_UB'):-diagnostico('PROVAVEL_FURO_LINHA').
acao('REDUZIR_CPM'):-diagnostico('PROVAVEL_PANCADA_FLUIDO'),
variavel('CPM',V),V>5.0.
acao('REDUZIR_CPM'):-diagnostico('ECO/IMPOSSIVEL_ESVAZIAR_TQ1'),
variavel('CPM',V),V>5.0,variavel('PRESSAO_FLUXO',V2),V2>0.1.
```

Base de conhecimento da UP ECO

a) Especificações da UP

```
cell.setDependenceType(RemoteCell.DEPENDENCE);
DBManager.getInstance().save(cell);

RemoteVariable rem = new RemoteVariable("VAZAO_BOMBA", cell);
DBManager.getInstance().save(rem);

cell = new RemoteCell("","ETO");
cell.setLocalVariable("NIVEL_TQ1");
cell.setRemoteVariable("NIVEL_TQ1");
cell.setTypeOfRelationship(RemoteCell.INVERSE);
cell.setDependenceType(RemoteCell.INFLUENCE);
DBManager.getInstance().save(cell);

rem = new RemoteVariable("NIVEL_TQ1", cell);
DBManager.getInstance().save(rem);

//NIVEL_TQ1
ProcessVariable var = new ProcessVariable("NIVEL_TQ1");
var.setTagName("[LEA]Program:MainProgram.EC_ACC");
var.setLimitCtMax(new RefValueDouble(new Double(1)));
var.setLimitCtMin(new RefValueDouble(new Double(0.5)));
var.setGlobal(true);
DBManager.getInstance().save(var);

var = new ProcessVariable("VAZAO_ENTRADA");
var.setTagName("[LEA]Program:MainProgram.EC_Vazao_in");
var.setLimitCtMax(new RefValueDouble(new Double(40)));
var.setLimitCtMin(new RefValueDouble(new Double(20)));
DBManager.getInstance().save(var);

var = new ProcessVariable("QTD_BOMBAS_ON");
var.setTagName("[LEA]Program:MainProgram.EC_qt_Bombas");
var.setDoubleValue(new Double(0));
DBManager.getInstance().save(var);

//BOMBA_TQ1
var = new ProcessVariable("BOMBA_1");
var.setDoubleValue(new Double(0));
var.setTagName("[LEA]Program:MainProgram.EC_Liga_M1");
DBManager.getInstance().save(var);

var = new ProcessVariable("BOMBA_2");
var.setDoubleValue(new Double(0));
var.setTagName("[LEA]Program:MainProgram.EC_Liga_M2");
DBManager.getInstance().save(var);

var = new ProcessVariable("BOMBA_3");
var.setDoubleValue(new Double(0));
var.setTagName("[LEA]Program:MainProgram.EC_Liga_M3");
DBManager.getInstance().save(var);

AdjustAction act = new AdjustAction("LIGAR_BOMBA_1");
act.setVariableName("BOMBA_1");
act.setGlobalInfluence(true);
act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);
```

```
act = new AdjustAction("LIGAR_BOMBA_2");
act.setVariableName("BOMBA_2");
act.setGlobalInfluence(true);
act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);

act = new AdjustAction("LIGAR_BOMBA_3");
act.setVariableName("BOMBA_3");
act.setGlobalInfluence(true);
act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_1");
act.setVariableName("BOMBA_1");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_2");
act.setVariableName("BOMBA_2");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_3");
act.setVariableName("BOMBA_3");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);
```

b) Regras

```
diagnostico('NIVEL_MEDIO_TANQUE'):-variavel('NIVEL_TQ1',V), V>0.5, V<0.8.
diagnostico('NIVEL_ALTO_TANQUE'):-variavel('NIVEL_TQ1',V), V>1.0.
diagnostico('NIVEL_MAXIMO_TANQUE'):-variavel('NIVEL_TQ1',V), V>1.8,
variavel('QTD_BOMBAS_ON',V), V=2.0.
diagnostico('NIVEL_BAIXO_TANQUE'):-variavel('NIVEL_TQ1',V), V<0.5.

%diagnostico('QUEDA_VAZAO_ENTRADA'):-variavel('VAZAO_ENTRADA',V), V<10.0.

diagnostico('IMPOSSIVEL_ESVAZIAR_TQ1'):-
diagnostico('ETO/IMPOSSIVEL_ESVAZIAR_TQ1'),diagnostico('NIVEL_ALTO_TANQUE').

acao('LIGAR_BOMBA_1'):-diagnostico('NIVEL_ALTO_TANQUE'),
variavel('QTD_BOMBAS_ON',QTD), QTD<1, variavel('BOMBA_1',B1), B1=0.0.
acao('LIGAR_BOMBA_2'):-diagnostico('NIVEL_ALTO_TANQUE'),
variavel('QTD_BOMBAS_ON',QTD), QTD=1.0, variavel('BOMBA_2',B2), B2=0.0.
acao('LIGAR_BOMBA_3'):-diagnostico('NIVEL_MAXIMO_TANQUE'),
variavel('BOMBA_3',B3), B3=0.0.
acao('LIGAR_BOMBA_3'):-diagnostico('NIVEL_ALTO_TANQUE'), diagnostico('PCO-
01/NIVEL_ALTO_ANULAR'), variavel('BOMBA_3',B3), B3=0.0.

acao('DESLIGAR_BOMBA_1'):-diagnostico('NIVEL_MEDIO_TANQUE'),
```

```
variavel('BOMBA_1',B1), B1=1.0; diagnostico('NIVEL_BAIXO_TANQUE'),
variavel('BOMBA_1',B1), B1=1.0.
acao('DESLIGAR_BOMBA_2'):-diagnostico('NIVEL_MEDIO_TANQUE'),
variavel('BOMBA_2',B2), variavel('BOMBA_1',B2), B2=1.0, B1=0.0;
diagnostico('NIVEL_BAIXO_TANQUE'), variavel('BOMBA_2',B2),
variavel('BOMBA_1',B2), B2=1.0, B1=0.0.
acao('DESLIGAR_BOMBA_3'):-diagnostico('NIVEL_BAIXO_TANQUE'),
variavel('BOMBA_3',B3), B3=1.0.

acao('DESLIGAR_BOMBA_1'):-diagnostico('ETO/IMPOSSIVEL_ESVAZIAR_TQ1'),
variavel('BOMBA_1',B1), B1=1.0.
acao('DESLIGAR_BOMBA_2'):-diagnostico('ETO/IMPOSSIVEL_ESVAZIAR_TQ1'),
variavel('BOMBA_2',B2), B2=1.0.
acao('DESLIGAR_BOMBA_3'):-diagnostico('ETO/IMPOSSIVEL_ESVAZIAR_TQ1'),
variavel('BOMBA_3',B3), B3=1.0.
```

Base de conhecimento da UP ETO

a) Especificações da UP

```
//Cells
RemoteCell cell = new RemoteCell("","ECO");
cell.setLocalVariable("NIVEL_TQ1");
cell.setRemoteVariable("NIVEL_TQ1");
cell.setTypeOfRelationship(RemoteCell.INVERSE);
DBManager.getInstance().save(cell);

RemoteVariable rem = new RemoteVariable("NIVEL_TQ1", cell);
DBManager.getInstance().save(rem);

//NIVEL_TQ1
ProcessVariable var = new ProcessVariable("NIVEL_TQ1");
var.setTagName("[ETO]Program:MainProgram.ET_Nivel");
var.setLimitCtMax(new RefValueDouble(new Double(1)));
var.setLimitCtMin(new RefValueDouble(new Double(0.5)));
var.setGlobal(true);
DBManager.getInstance().save(var);

var = new ProcessVariable("QTD_BOMBAS_ON");
var.setTagName("[ETO]Program:MainProgram.EC_qt_Bombas");
var.setDoubleValue(new Double(0));
DBManager.getInstance().save(var);

//BOMBA_TQ1
var = new ProcessVariable("BOMBA_1");
var.setDoubleValue(new Double(0));
var.setTagName("[ETO]Program:MainProgram.EC_Liga_M1");
DBManager.getInstance().save(var);

var = new ProcessVariable("BOMBA_2");
var.setDoubleValue(new Double(0));
var.setTagName("[ETO]Program:MainProgram.EC_Liga_M2");
DBManager.getInstance().save(var);

var = new ProcessVariable("BOMBA_3");
```

```
var.setDoubleValue(new Double(0));
var.setTagName("[ETO]Program:MainProgram.EC_Liga_M3");
DBManager.getInstance().save(var);

AdjustAction act = new AdjustAction("LIGAR_BOMBA_1");
act.setVariableName("BOMBA_1");
act.setGlobalInfluence(false);
//act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);

act = new AdjustAction("LIGAR_BOMBA_2");
act.setVariableName("BOMBA_2");
act.setGlobalInfluence(false);
//act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);

act = new AdjustAction("LIGAR_BOMBA_3");
act.setVariableName("BOMBA_3");
act.setGlobalInfluence(false);
//act.setGlobalVariable("NIVEL_TQ1");
act.setAdjustType(AdjustAction.AT_DECREASE);
act.setNewValue(new RefValueDouble(new Double(1)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_1");
act.setVariableName("BOMBA_1");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_2");
act.setVariableName("BOMBA_2");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);

act = new AdjustAction("DESLIGAR_BOMBA_3");
act.setVariableName("BOMBA_3");
act.setGlobalInfluence(false);
act.setNewValue(new RefValueDouble(new Double(0)));
DBManager.getInstance().save(act);
```

b) Regras

```
diagnostico('NIVEL_MEDIO_TANQUE'):-variavel('NIVEL_TQ1',V), V>0.5, V<0.8.
diagnostico('NIVEL_ALTO_TANQUE'):-variavel('NIVEL_TQ1',V), V>1.0.
diagnostico('NIVEL_MAXIMO_TANQUE'):-variavel('NIVEL_TQ1',V), V>1.4.
diagnostico('NIVEL_BAIXO_TANQUE'):-variavel('NIVEL_TQ1',V), V<0.5.

diagnostico('IMPOSSIVEL_ESVAZIAR_TQ1'):-
diagnostico('NIVEL_MAXIMO_TANQUE'),variavel('QTD_BOMBAS_ON',V),V>=2.0.

acao('LIGAR_BOMBA_1'):-diagnostico('NIVEL_ALTO_TANQUE'),
variavel('QTD_BOMBAS_ON',QTD), QTD<1, variavel('BOMBA_1',B1), B1=0.0.
acao('LIGAR_BOMBA_2'):-diagnostico('NIVEL_ALTO_TANQUE'),
```

```
variavel('QTD_BOMBAS_ON',QTD), QTD=1.0, variavel('BOMBA_2',B2), B2=0.0.
acao('LIGAR_BOMBA_3'):-diagnostico('NIVEL_MAXIMO_TANQUE'),
variavel('BOMBA_3',B3), B3=0.0.
acao('LIGAR_BOMBA_3'):-diagnostico('ECO/NIVEL_MAXIMO_TANQUE'),
variavel('BOMBA_3',B3), B3=0.0.

acao('DESLIGAR_BOMBA_1'):-diagnostico('NIVEL_MEDIO_TANQUE');
diagnostico('NIVEL_BAIXO_TANQUE'), variavel('BOMBA_1',B1), B1=1.0.
acao('DESLIGAR_BOMBA_2'):-diagnostico('NIVEL_MEDIO_TANQUE');
diagnostico('NIVEL_BAIXO_TANQUE'), variavel('BOMBA_2',B2),
variavel('BOMBA_1',B2), B2=1.0, B1=0.0.
acao('DESLIGAR_BOMBA_3'):-diagnostico('NIVEL_BAIXO_TANQUE'),
variavel('BOMBA_3',B3), B3=1.0.
```

APÊNDICE F – DOCUMENTAÇÃO DOS AGENTES

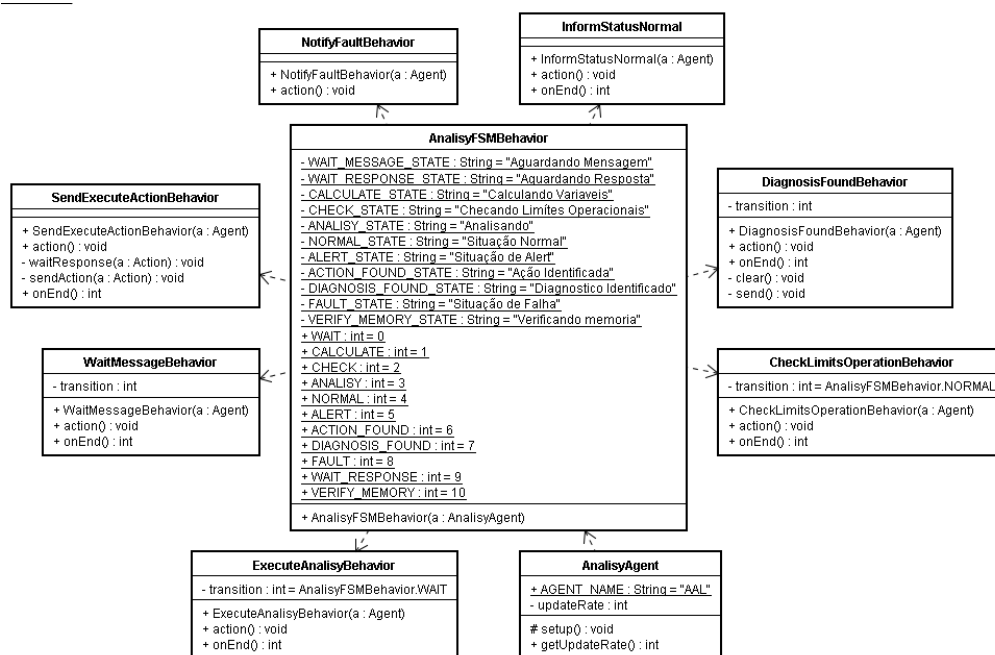



Figura 46: Diagrama de classes do AAL

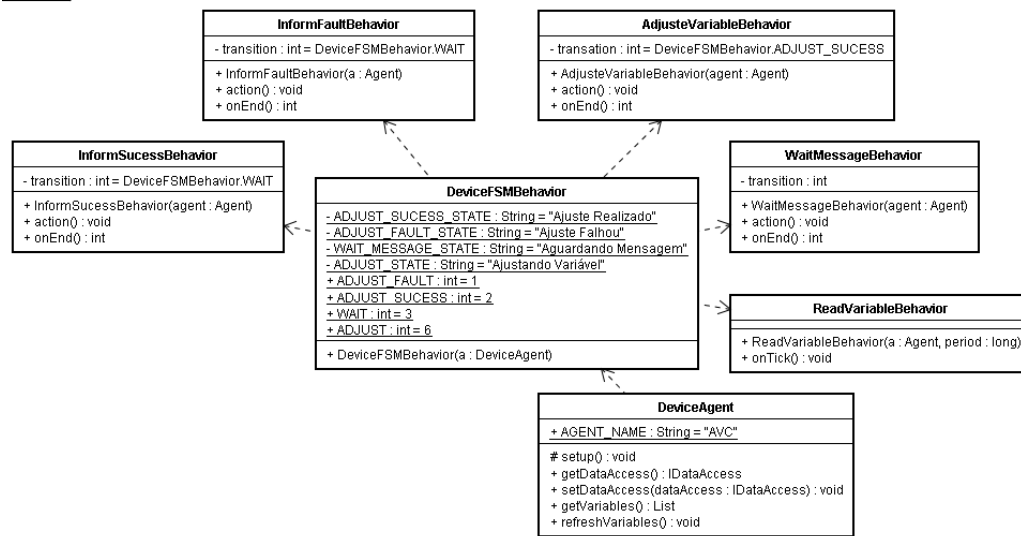



Figura 47: Diagrama de classes do AVC

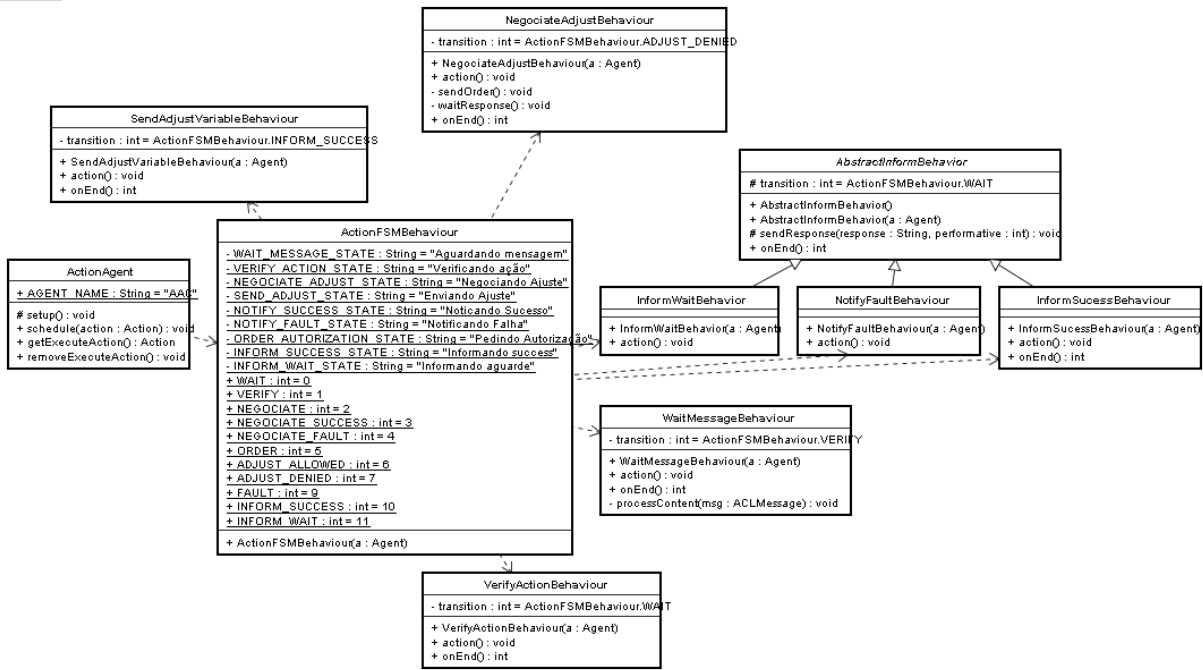



Figura 48: Diagrama de classes do AAC

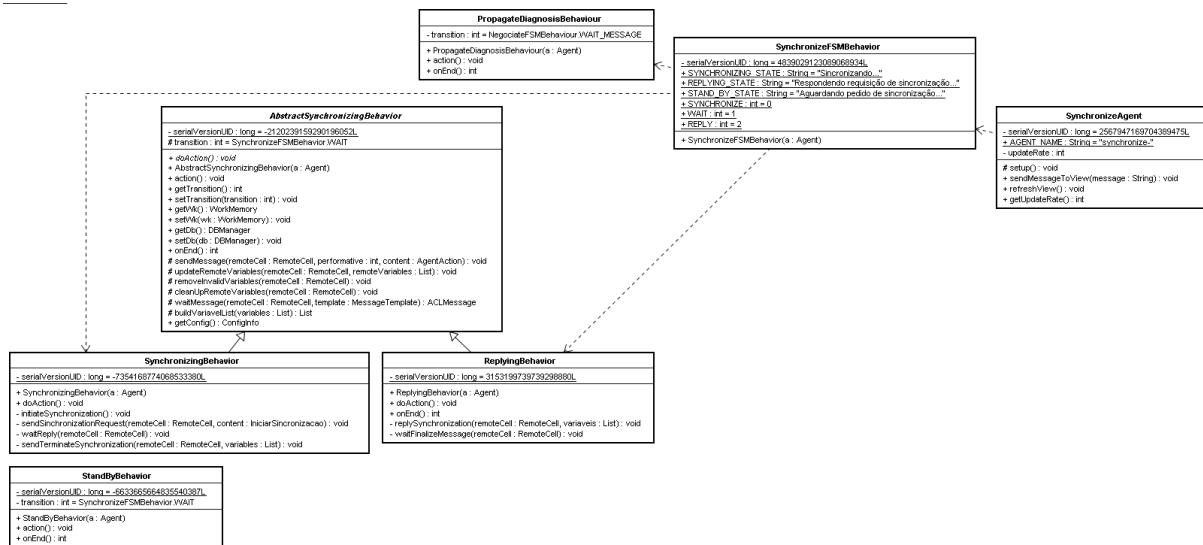



Figura 49: Diagrama de classes do ASI

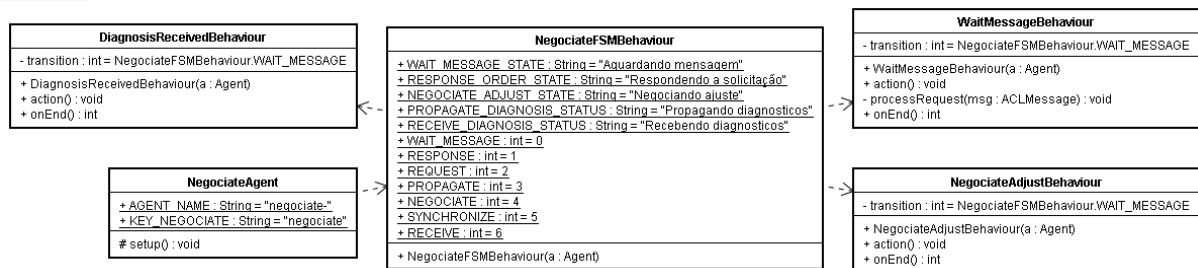



Figura 50: Diagrama de classes do ANE